ICON-2017

14th International Conference on Natural Language Processing

Proceedings of the Conference

18-21 December 2017
Jadavpur University, Kolkata, India

Preface

Research in Natural Language Processing (NLP) has taken a noticeable leap in the recent years. Tremendous growth of information on the web and its easy access has stimulated large interest in the field. India with multiple languages and continuous growth of Indian language content on the web makes a fertile ground for NLP research. Moreover, industry is keenly interested in obtaining NLP technology for mass use. The internet search companies are increasingly aware of the large market for processing languages other than English. For example, search capability is needed for content in Indian and other languages. There is also a need for searching content in multiple languages, and making the retrieved documents available in the language of the user. As a result, a strong need is being felt for machine translation to handle this large instantaneous use. Information Extraction, Question Answering Systems and Sentiment Analysis are also showing up as other business opportunities.

These needs have resulted in two welcome trends. First, there is much wider student interest in getting into NLP at both postgraduate and undergraduate levels. Many students interested in computing technology are getting interested in natural language technology, and those interested in pursuing computing research are joining NLP research. Second, the research community in academic institutions and the government funding agencies in India have joined hands to launch consortia projects to develop NLP products. Each consortium project is a multi-institutional endeavour working with a common software framework, common language standards, and common technology engines for all the different languages covered in the consortium. As a result, it has already led to development of basic tools for multiple languages which are inter-operable for machine translation, cross lingual search, hand writing recognition and OCR.

In this backdrop of increased student interest, greater funding and most importantly, common standards and interoperable tools, there has been a spurt in research in NLP on Indian languages whose effects we have just begun to see. A great number of submissions reflecting good research is a heartening matter. There is an increasing realization to take advantage of features common to Indian languages in machine learning. It is a delight to see that such features are not just specific to Indian languages but to a large number of languages of the world, hitherto ignored. The insights so gained are furthering our linguistic understanding and will help in technology development for hopefully all languages of the world.

For machine learning and other purposes, linguistically annotated corpora using the common standards have become available for multiple Indian languages. They have been used for the development of basic technologies for several languages. Larger set of corpora are expected to be prepared in near future.

This conference proceedings contains papers selected for presentation in technical sessions of ICON-2017 and short communications selected for poster presentation. We are thankful to our excellent team of reviewers from all over the globe who deserve full credit for the hard work of reviewing the high quality submissions with rich technical content. From 141 submissions, 64 papers were selected, 32 for full presentation, 32 for poster presentation, representing a variety of new and interesting developments, covering a wide spectrum of NLP areas and core linguistics.

We are deeply grateful to Bjrn W. Schuller, University of Passau, Germany, NG Hwee Tou, National University of Singapore (NUS), Singapore and Vasudeva Varma, IIIT Hyderabad, India for giving the keynote lectures at ICON. We would also like to thank the members of the Advisory Committee and

Programme Committee for their support and co-operation in making ICON 2017 a success.

We thank Anil Kumar Singh, Chair, Student Paper Competition and Dipankar Das, Chair, NLP Tools Contest for taking the responsibilities of the events. We are thankful to Sudip Kumar Naskar and Dipankar Das for making the organization of the event at Jadavpur University a success.

We convey our thanks to P V S Ram Babu, G Srinivas Rao, B Mahender Kumar and A Lakshmi Narayana, International Institute of Information Technology (IIIT), Hyderabad for their dedicated efforts in successfully handling the ICON Secretariat. We also thank IIIT Hyderabad team of Vineet Chaitanya, Vasudeva Varma, Soma Paul, Radhika Mamidi, Manish Shrivastava, Suryakanth V Gangashetty and Anil Kumar Vuppala. We heartfully express our gratitude to Somnath Banerjee, Tapabrata Mondal, Sainik Mahata and other team members at Jadavpur University for their timely help with sincere dedication to make this conference a success.

We also thank all those who came forward to help us in this task.

Finally, we thank all the researchers who responded to our call for papers and all the participants of ICON-2017, without whose overwhelming response the conference would not have been a success.

December 2017
Varanasi

Sivaji Bandyopadhyay
Dipti Misra Sharma
Rajeev Sangal

Advisory Committee:

Aravind K Joshi, University of Pennsylvania, USA (Chair)

Conference General Chair:

Rajeev Sangal, IIT (BHU), Varanasi, India

Programme Committee:

Sivaji Bandyopadhyay, Jadavpur University,Kolkata, India (Chair)
Dipti Misra Sharma, IIIT Hyderabad, India (Co-Chair)
Kalika Bali, Microsoft Research India, India
Srinivas Bangalore, Interactions LLC, AT&T Research, USA
Rajesh Bhatt, University of Massachusetts, USA
Pushpak Bhattacharyya, IIT Patna, India
Monojit Choudhury, Microsoft Research India, India
Josef van Genabith, DFKI GmbH, Germany
Harald Hammarstrm, Max Planck Institute for Psycholinguistics, Nijmegen, The Netherlands
Mohammed Hasanuzzaman, Universit de Caen, Normandie, France
Gurpreet Lehal, Punjabi University, Patiala, India
Yuji Matsumoto, Nara Institute of Science and Technology, Japan
Joakim Nivre, Uppsala University, Sweden
Alexis Palmer, Heidelberg University, Germany
Jyoti Pawar, DCST, Goa University, India
Owen Rambow, University of Columbia, USA
Paolo Rosso, Universitat Politcnica de Valncia, Spain
Shikhar Kr. Sarma, Gauhati University, India
Elizabeth Sherly, IIITM-K, Trivandrum, India
Sobha Lalitha Devi, AU-KBC, Chennai, India
Keh-Yih Su, Institute of Information Science, Academia Sinica, Taiwan
Vasudeva Varma, IIIT Hyderabad, India

Tools Contest Chairs:

Dipankar Das, Jadavpur University, Kolkata, India (Chair)
Amitava Das, IIIT-Sri City, India

Student Paper Competition Chair:

Anil Kumar Singh, IIT (BHU), India

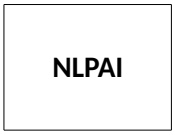
Organizing Committee:

Sudip Naskar, Jadavpur University, Kolkata, India
Dipankar Das, Jadavpur University, Kolkata, India

Organized by:

Jadavpur University

IIIT-H



NLPAI

LDC-IL, CIIL Mysore

Sponsored by:

Microsoft Research India

American Express

Referees

We gratefully acknowledge the excellent quality of refereeing we received from the reviewers. We thank them all for being precise and fair in their assessment and for reviewing the papers in time.

Abhijeet Mishra
Adithya Pratapa
Aditi Mukherjee
Aditya Joshi
Aishwarya N Reganti
Akhilesh Sudhakar
Alok Ranjan Pal
Amitava Das
Amol Bole
Anand Kumar
Anchal Rani
Anil Kumar Singh
Anil Thakur
Anil Kumar Vuppala
Animesh Mukherjee
Anshul Bawa
Anupam Jamatia
Anupam Mondal
Anusha Prakash
Anushiya Rachel G
Arjun Akula
Arun Baby
Ashika Naidu
Ashwini Vaidya
Asif Ekbal
Aswin Shanmugam S
Bajibabu Bollepalli
Balamurali A R
Bapi Raju
Basil Abraham
Bharat Ram Ambati
Bhuvana Narasimhan
Braja Gopal Patra
C V Jawahar
Chandra Sekhar Chellu
Christian Boitet
D S Karthik Pandia
Dan Zeman
Deepak P
Delia Irazú Hernández Farias
Devi G
Dipankar Das
Dipti Misra Sharma
Divya Sai Jitta
Dwijen Rudrapal
Elizabeth Sherly
Erik Cambria
Fei Xia
Girish Palshikar
Golda Brunet Rajan
Gurpreet Singh Lehal
Harald Hammarström
Hema Murthy
Himanshu Sharma
Irshad Bhat
Jilt Sebastian
Jose Moreno
Joy Mahapatra
Jyoti Pawar
Kalika Bali
Kamal Sarkar
Karunesh Arora
Keh-Yih Su
Kevin Patel
Kishorjit Nongmeikapam
Kunal Chakma
Litton J Kurisinkel
Mahidas Bhattacharya
Malhar Kulkarni
Manish Shrivastava
Manoj Chinnakotla
Maria Anzovino
Mayank Singh
Mohammed Hasanuzzaman
Monojit Choudhury
Nikhil Pattisapu
Niloofar Safi
Niraj Kumar
Nirmal Surange
Owen Rambow
Paolo Rosso
Parameswari Krishnamurthy
Partha Pakray
Partha Talukdar
Pawan Goyal
Pranav Dhakras

Pranav Goel
Pranaw Kumar
Prathyusha Jwalapuram
Priya Radhakrishnan
Pruthwik Mishra
Pushpak Bhattacharyya
Radhika Mamidi
Rajeev Rajan
Rajiv Srivastava
Rakesh Balabantaray
Ranjani Parthasarathi
Rashmi Prasad
Riyaz Bhat
Rudramurthy V
Sachin Pawar
Sakshi Kalra
Samar Husain
Samudravijaya K
Sanjukta Ghosh
Santanu Pal
Satarupa Guha
Shashank Gupta
Shashi Narayan
Shilpa Desai
Siva Reddy Gangireddy
Sivaji Bandyopadhyay
Sivanand Achanta
Smriti Singh
Sobha L
Soma Paul
Somnath Banerjee
Sourav Mandal
Spandana Gella
Sri Rama Murty Kodukula
Srikanth Ronanki
Sriram Venkatapathy
Sruti Rallapalli
Subba Reddy Oota
Subhabrata Dutta
Sudip Kumar Naskar
Sudipta Kar
Sutanu Chakraborti
Swapnil Chaudhari
Swapnil Hingmire
T Nagarajan
Thamar Solorio
Thoudam Doren Singh
Tushar Maheshwari
Umamaheswari E
Vasudeva Varma
Vasudevan Nedumpozhi
Venkata Viraraghavan
Vijay Sundar Ram
Vijayalakshmi P
Vinay Kumar Mittal
Vishal Goyal
Yan Shao
Yuji Matsumoto

Table of Contents

Conference Program

Tuesday, December 19, 2017

+ 9:00-9:30 Inaugural Ceremony

+ 9:30-10:30 Keynote Lecture 1 by Prof. Bjom W. Schuller

Keynote Lecture 1: NLP in Tomorrow's Profiling - Words May Fail You
Björn W. Schuller

+ 10:30-11:00 Tea Break

+ 11:00-13:00 Technical Session I: Machine Translation and Speech:

Deriving Word Prosody from Orthography in Hindi
Somnath Roy

Three-phase training to address data sparsity in Neural Machine Translation
Ruchit Agrawal, Mihir Shekhar and Dipti Sharma

Reference Scope Identification for Citances Using Convolutional Neural Networks
Saurav Jha, Aanchal Chaurasia, Akhilesh Sudhakar and Anil Kumar Singh

A vis-à-vis evaluation of MT paradigms for linguistically distant languages
Ruchit Agrawal, Jahfar Ali and Dipti Misra Sharma

+ 11:00-13:00 Technical Session II : Text Categorization:

Textual Relations and Topic-Projection: Issues in Text Categorization
Lahari Chatterjee, Samir Karmakar and Abahan Datta

POS Tagging For Resource Poor Languages Through Feature Projection
Pruthwik Mishra, Vandan Mujadia and Dipti Misra Sharma

An Exploration of Word Embedding Initialization in Deep-Learning Tasks
Tom Kocmi and Ondrej Bojar

Tuesday, December 19, 2017 (continued)

+ 11:00-13:00 Technical Session III : Parsing Code-mixed Data:

Curriculum Design for Code-switching: Experiments with Language Identification and Language Modeling with Deep Neural Networks
Monojit Choudhury, Kalika Bali, Sunayana Sitaram and Ashutosh Baheti

Quantitative Characterization of Code Switching Patterns in Complex Multi-Party Conversations: A Case Study on Hindi Movie Scripts
Adithya Pratapa and Monojit Choudhury

Towards Normalising Konkani-English Code-Mixed Social Media Text
Akshata Phadte and Gaurish Thakkar

Towards developing a phonetically balanced code-mixed speech corpus for Hindi-English ASR
Ayushi Pandey, Brij Mohan Lal Srivastava and Suryakanth Gangashetty

+ 13:00-14:00 Lunch

+ 14:00-15:00 Keynote Lecture 2 by Prof. NG Hwee Tou

Keynote Lecture 2: Grammatical Error Correction: Past, Present and Future
NG Hwee Tou

+ 15:00-16:30 Technical Session IV : Information Extraction:

Hybrid Approach for Marathi Named Entity Recognition
Nita Patil, Ajay Patil and B.V. Pawar

Sentiment Analysis: An Empirical Comparative Study of Various Machine Learning Approaches
Swapnil Jain, Shrikant Malviya, Rohit Mishra and Uma Shanker Tiwary

Tuesday, December 19, 2017 (continued)

+ 15:00-16:30 Technical Session V : Discourse and Dialogue:

Handling Multi-Sentence Queries in a Domain Independent Dialogue System
Prathyusha Jwalapuram and Radhika Mamidi

Document Level Novelty Detection: Textual Entailment Lends a Helping Hand
Tanik Saikh, Tirthankar Ghosal, Asif Ekbal and Pushpak Bhattacharyya

Is your Statement Purposeless? Predicting Computer Science Graduation Admission Acceptance based on Statement Of Purpose
Diptesh Kanojia, Nikhil Wani and Pushpak Bhattacharyya

+ 15:00-16:30 Technical Session VI : Lexical Analysis:

Natural Language Programing with Automatic Code Generation towards Solving Addition-Subtraction Word Problems
Sourav Mandal and Sudip Kumar Naskar

Unsupervised Separation of Transliterable and Native Words for Malayalam
Deepak P

Known Strangers: Cross Linguistic Patterns in Multilingual Multidirectional Dictionaries

Rejitha K. S. and Rajesha N.

+ 16:30-17:30 Tea Break

+ 16:30-17:30 Poster and Demo Session-I:

Tutorial for Deaf – Teaching Punjabi Alphabet using Synthetic Animations
Lalit Goyal and Vishal Goyal

SemTagger: A Novel Approach for Semantic Similarity Based Hashtag Recommendation on Twitter
Kuntal Dey, Ritvik Shrivastava, Saroj Kaushik and L. Venkata Subramaniam

Reasoning with Sets to Solve Simple Word Problems Automatically
Sowmya S Sundaram and Deepak Khemani

Improving NER for Clinical Texts by Ensemble Approach using Segment Representations
Hamada Nayel and H L Shashirekha

Tuesday, December 19, 2017 (continued)

Beyond Word2Vec: Embedding Words and Phrases in Same Vector Space
Vijay Prakash Dwivedi and Manish Shrivastava

Relationship Extraction based on Category of Medical Concepts from Lexical Contexts
Anupam Mondal, Dipankar Das and Sivaji Bandyopadhyay

Sinhala Word Joiner
Rajith Priyanga, Surangika Ranatunga and Gihan Dias

Supervised Methods For Ranking Relations In Web Search
Sumit Asthana and Asif Ekbal

Malayalam VerbFrames
Jisha P Jayan, Asha S Nair and Govindaru V

Hindi Shabdamitra: A Wordnet based E-Learning Tool for Language Learning and Teaching
Hanumant Redkar, Sandhya Singh, Dhara Gorasia, Meenakshi Somasundaram, Malhar Kulkarni and Pushpak Bhattacharyya

"A pessimist sees the difficulty in every opportunity; an optimist sees the opportunity in every difficulty" – Understanding the psycho-sociological influences to it
Updendra Kumar, Vishal Kumar Rana, Srinivas Pykl and Amitava Das

End to End Dialog System for Telugu
Prathyusha Danda, Prathyusha Jwalapuram and Manish Shrivastava

Investigating how well contextual features are captured by bi-directional recurrent neural network models
Kushal Chawla, Sunil Kumar Sahu and Ashish Anand

Correcting General Purpose ASR Errors using Posteriors
Sunil Kumar Kopparapu and C. Anantaram

Retrieving Similar Lyrics for Music Recommendation System
Braja Gopal Patra, Dipankar Das and Sivaji Bandyopadhyay

Unsupervised Morpheme Segmentation Through Numerical Weighting and Thresholding
Joy Mahapatra and Sudip Kumar Naskar

Tuesday, December 19, 2017 (continued)

Experiments with Domain Dependent Dialogue Act Classification using Open-Domain Dialogue Corpora
Swapnil Hingmire, Apoorv Shrivastava, Girish Palshikar and Saurabh Srivastava

Normalization of Social Media Text using Deep Neural Networks
Ajay Shankar Tiwari and Sudip Kumar Naskar

Acronym Expansion: A General Approach Using Deep Learning
Aditya Thakker, Suhail Barot and Sudhir Bagul

Exploring an Efficient Handwritten Manipuri Meetei-Mayek Character Recognition Using Gradient Feature Extractor and Cosine Distance Based Multiclass k-Nearest Neighbor Classifier
Kishorjit Nongmeikapam, Wahengbam Kumar and Mithlesh Prasad Singh

A Modified Cosine-Similarity based Log Kernel for Support Vector Machines in the Domain of Text Classification
Rajendra Kumar Roul and Kushagr Arora

Document Embedding Generation for Cyber-Aggressive Comment Detection using Supervised Machine Learning Approach
Shylaja S S, Abhishek Narayanan, Abhijith Venugopal and Abhishek Prasad

Coarticulatory propensity in Khalkha Mongolian
Ushashi Banerjee, Indranil Dutta and Irfan S

Developing Lexicon and Classifier for Personality Identification in Texts
Kumar Gourav Das and Dipankar Das

Linguistic approach based Transfer Learning for Sentiment Classification in Hindi
Vartika Rai, Sakshee Vijay and Dipti Misra

Scalable Bio-Molecular Event Extraction System towards Knowledge Acquisition
Pattabhi Rk Rao, Sindhuja Gopalan and Sobha Lalitha Devi

Co-reference Resolution in Tamil Text
Vijay Sundar Ram and Sobha Lalitha Devi

Cross Linguistic Variations in Discourse Relations among Indian Languages
Sindhuja Gopalan, Lakshmi S and Sobha Lalitha Devi

Tuesday, December 19, 2017 (continued)

RULE BASED APPROCH OF CLAUSE BOUNDARY IDENTI-FICATION IN TELUGU
Ganthoti Nagaraju, Thennarasu Sakkan and Christopher Mala

+ 17:30-19:30 NLPAI Meeting

+ 19:00-20:00 Cultural Programme

+ 20:00-Onwards Dinner

Wednesday, December 20, 2017

+ 9:30-10:30 Keynote Lecture 3 by Vasudeva Varmaa

Keynote Lecture 3: Towards Abstractive Summarization
Vasudeva Varma

+ 10:30-11:00 Tea Break

+ 11:00-13:00 Technical Session VII: Socio-Psycho Text Analysis: Emerging Trends

"Who Mentions Whom?"- Understanding the Psycho-Sociological Aspects of Twitter Mention Network
R Sudhesh Solomon, Abhay Narayan, Srinivas P Y K L and Amitava Das

Study on Visual Word Recognition in Bangla across Different Reader Groups
Manjira Sinha and Tirthankar Dasgupta

Demystifying Topology of Autopilot Thoughts: A Computational Analysis of Linguistic Patterns of Psychological Aspects in Mental Health
Bibekananda Kundu and Sanjay Choudhury

A Deep Dive into Identification of Characters from Mahabharata
Apurba Paul and Dipankar Das

Wednesday, December 20, 2017 (continued)

+ 11:00-13:00 Technical Session VIII: Deep Neural Networks:

Neural Networks for Semantic Textual Similarity
Derek Prijatelj, Jugal Kalita and Jonathan Ventura

Open Set Text Classification using Convolutional Neural Networks
Sridhama Prakhya, Vinodini Venkataram and Jugal Kalita

Predicting User Competence from Linguistic Data
Yonas Woldemariam, Henrik Björklund and Suna Bensch

Neural Morphological Disambiguation Using Surface and Contextual Morphological Awareness
Akhilesh Sudhakar and Anil Kumar Singh

+ 11:00-13:00 Technical Session IX: Semantics:

Word Sense Disambiguation for Malayalam in a Conditional Random Field Framework
Junaida M K, Jisha P Jayan and Elizabeth Sherly

Semisupervied Data Driven Word Sense Disambiguation for Resource-poor Languages
Pratibha Rani, Vikram Pudi and Dipti M. Sharma

Notion of Semantics in Computer Science - A Systematic Literature Review
Sai Prasad Vrj Gollapudi and Venkatesh Choppella

Semantic Enrichment Across Language: A Case Study of Czech Bibliographic Databases
Pavel Smrz and Lubomir Otrusina

Wednesday, December 20, 2017 (continued)

+ 13:00-14:00 Lunch Break

+ 14:00-15:00 Industry Talk

+ 15:00-15:30 Tea Break

+ 15:30-17:00 Technical Session X: Student Paper Contest

+ 15:30-17:00 Technical Session XI: NLP Tools Contest

+ 17:00-17:30 Valedictory Session

Keynote Lecture-1

NLP in Tomorrow's Profiling - Words May Fail You

Björn W. Schuller
University of Augsburg, Germany

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, page 1

Deriving Word Prosody from Orthography in Hindi

Somnath Roy
Centre for Linguistics
Jawaharlal Nehru University
New Delhi-110067
somnathroy86@gmail.com

Abstract

This study proposes a word prosody converter (WPC), which takes Hindi grapheme as input and yields output as a sequence of phonemes with syllable boundaries and stress mark. The WPC has two submodules connected in the linear fashion. The first submodule is a grapheme to phoneme (G2P) converter. The output of G2P converter is fed to the second submodule which is for prosody specific job. The second submodule consists of two finite state machines (FSMs). The first FSM does the syllabification and the second assigns prosodic labels to the syllabified strings. The prosodic labels are translated into the stressed and unstressed component using rules specific to the language. This study proposes a novel rule-based system which uses non-linear phonological rules with the provision of recursive foot structure for G2P conversion and prosodic labeling. The implementation[1] of the proposed rules outperforms the G2P models trained on the state of the art data-driven techniques such as joint sequence model (JSM) and LSTM.

1 Introduction

A dictionary is an essential component of a text-to-speech (TTS) and an automatic speech recognition (ASR) system. These systems are of open nature and can have an input word which is not present in the dictionary. Such input words are called out-of-vocabulary (OOV) words. Therefore, a G2P converter is required, which can generate the pronunciation of the OOV words. A G2P converter can be a rule-based or data driven system. A rule-based G2P converter relies on the expert knowledge (i.e., the rule-set designed by an expert). However, these rule-sets may not be exhaustive for capturing many language-specific properties such as word morphology and stress pattern (Pagel et al., 1998).Therefore, researchers nowadays rely on state-of-the art machine learning (data-driven) techniques for developing a G2P model. A data-driven system is trained using a manually annotated dataset. The manually annotated dataset contains words and its phonemic sequence. These datasets are language specific in nature. The machine learning algorithm learns the phonemic sequence for words based on the probabilistic or geometric calculation. These calculation varies across machine learning approaches. In data-driven approaches, one need not to worry about the language specific complexities such as word morphology and stress pattern. The algorithm automatically captures these patterns in the generated model. A data-driven G2P conversion process is broadly categorized into three subprocesses i) Sequence alignment ii) Model training and iii) Decoding (for details see (Novak et al., 2012)). Many data-driven techniques are available for G2P conversion. The important ones are decision tree (Black et al., 1998), Conditional Random Field (Wang and King, 2011), Hidden Markov Model (Taylor, 2005), Joint-Sequence techniques (Bisani and Ney, 2008) and Recurrent Neural Network (Rao et al., 2015).

The function of a word prosody model is similar to that of a grapheme to phoneme (G2P) converter. Moreover, it also describes syllable boundaries and predict stressed syllables in a word. The schematic diagram of word prosody model is shown in Fig 1. The accuracy of a word prosody module for Hindi language depends on an efficient solution of the two sub-problems well-known in Hindi phonology as schwa deletion and pronunciation of diacritic marks anusvara and anunasika (Ohala, 1983; Pandey, 1989; Pandey,

[1] https://github.com/somnat/Hindi-Word-Prosody-Hindi-G2P

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 2–12

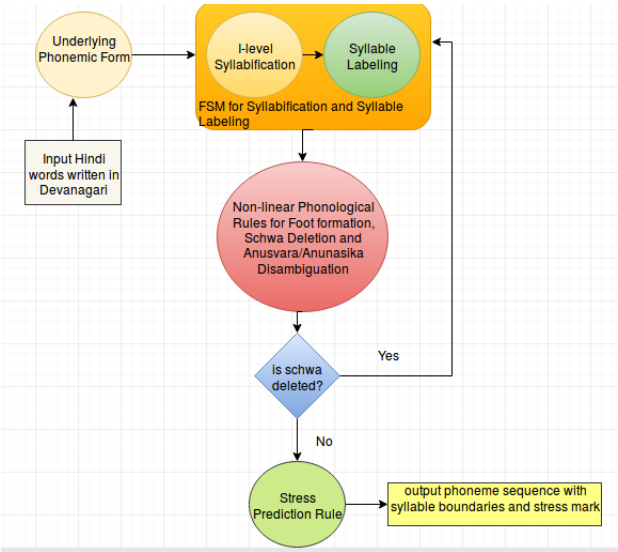


Figure 1: Schematic Diagram of Word Prosody Model

1990; Narasimhan et al., 2004; Pandey, 2014). Ohala used linear phonological rules to derive surface phonemic form. Pandey showed the superiority of non-linear phonological rules over the linear one. The motivation for the current work is stated below.

a. In the past, Hindi G2P converters were implemented in the context of speech synthesis (Bali et al., 2004), (Narasimhan et al., 2004) and (Choudhury, 2003). However, these works have given partial attention to the anusvara/anunasika disambiguation. (Pandey, 2014) describes it as the problem of Hindi orthography.

b. These G2P converters are based on linear phonological rules proposed by (Ohala, 1983) with the exception of (Pandey, 2014). Non-linear phonological rules have advantages over linear one as explained below. (Bernhardt and Gilbert, 1992).

i. Non-linear rules capture both the prosodic and segmental information.

ii. The hierarchical representation used in non-linear framework captures more information; this results in a compact rule set.

c. Syllable is known to be a better unit for Hindi speech synthesis (Bellur et al., 2011; Kishore and Black, 2003). Therefore, a Hindi text-to-speech (TTS) system needs an automatic syllabification module. The automatic syllabification would be more useful if it could also predict the stressed syllables in words of natural speech as this would facilitate synthesis.

d. The usefulness of syllable as the basic linguistic unit in the context of speech recognition system has been explored in English (Ganapathiraju et al., 2001) and Tamil (Lakshmi and Murthy, 2006). Similar work for Hindi requires a software for syllabification. This work fulfills that need.

1.1 Main Contributions

- The WPC does not require the information of morphological boundaries. The proposed rules take into account the syllable patterns of compound, derived and inflected words.

- The syllabification and syllable labeling process follow finite state machine. The faultless syllabification and syllable labeling at underlying phonemic form yields better accuracy in schwa deletion and pronunciation of diacritic—anusvara and anunasika. The syllabification at underlying phonemic form is called as I-level syllabification in this work.

- The rules proposed in this study assume the extrametricality of foot unlike syllable as proposed in (Pandey, 2014). The contention is that the stress can be predicted elegantly using the notion of extrametrical foot (McCarthy and Prince, 1990; Crowhurst, 1994) . Also, the directionality is LR (left to right) unlike RL (right to left) used in (Pandey, 2014).

- Anusvara and anunasika are used interchangeably in Hindi. Therefore, both anusvara and anunasika is mapped to a hypothetical phoneme X at the underlying phonemic form. The decision for homo-organic nasal consonant or a nasalized vowel for phoneme X is based on the minimum moraic weight difference of the syllable having phoneme X and the next syllable. The moraic weight difference is calculated after schwa deletion and re-syllabification. The proposed mapping rule almost removes the pronunciation ambiguity related to anusvara and anunasika.

Rest of this paper is organized as follows. Section 2 describes the salient points of metrical phonology relevant to this work. Section 3 describes the process of syllabification and syllable labeling. Section 4 describes foot formation. Section 5 describes schwa deletion and re-syllabification. Section 6 describes the observa-

tions and rules for the anusvara and anunasika pronunciation. Section 7 describes the data-driven G2P systems implemented for Hindi. Section 8 compares the performance of current system to data-driven systems and previous rule-based implementations. Section 9 describes the rules for the prediction of the stressed syllables and reports the accuracy of current system for syllabification and stress prediction. The conclusion and limitations are written in Section 10.

2 Theoretical Background

Metrical phonology is based on nonlinear arrangement of the constituents of a phrase (Liberman and Prince, 1977; Selkirk, 1980; Hayes, 1980; Selkirk, 1986; Hayes, 1995; Apoussidou, 2006). The nonlinear arrangement is realized in the form of a tree with nodes as the constituents of a phrase. The constituents are syllable, foot, phonological word, phonological phrase and intonational phrase. Syllable is the lowest unit in the hierarchy dominated by foot, which in turn is dominated by a phonological word. The higher units such as phonological phrase and intonational phrase are not relevant in the current work (for clarity see fig 4 - 9). Syllable functions as a domain for segmental phonological rules. In non-linear phonology, the rules are written on the basis of interaction among syllables under the domain of higher constituents. A syllable has obligatory rhyme and optional coda. The syllables are also described by the moraic weight in quantity-sensitive languages such as Hindi (Pandey, 1989). Foot as a domain is used for describing stress and re-syllabification due to deletion of segment like schwa in languages such as French and Hindi. The foot is used as a musical meter and the concept is borrowed to non-linear phonology as a constituent (Selkirk, 1980; Hayes, 1995). Most of the quantity sensitive languages have binary foot, but some also allow degenerate foot. A binary foot is erected on either two syllables or on a single syllable having at least two moras. A single syllable having one mora, if projected as a foot, is called degenerate foot (Liberman and Prince, 1977; Hayes, 1995).Phonological word, also known as prosodic word, is a constituent unit of prosodic hierarchy above syllables or foot and below phonological phrases. Prosodic word is non-isomorphic to the grammatical word and the boundary of the former aligns with the morpho-syntactic boundary (Hall and Kleinhenz, 1999).

3 I-Level Syllabification

I-level syllabification is derived from the underlying phonemic form (UPF), which in turn is derived from orthography using the following mapping rules.

i. Each consonant in Devanagari script is inherently associated with the mid-central vowel called schwa or its lower counterpart "a"[2].

ii. If a consonant is followed by a vowel diacritic mark, or a diacritic called halant, the inherent schwa is deleted.

iii. The inherent schwa is not realized in case of consonant at word final position.

iv. Two or three consonant together can form a ligature.

v. A short vowel at word final position is lengthened.

The following examples illustrate derivation of UPF from orthography:

/kml/ → kəməl (Lotus)

/kmAl/ → kəmaːl

The process of syllabification in Hindi was explored by (Ohala, 1983) and (Pandey, 1989; Pandey, 2014). Their analysis do not talk about the maximal onset principle for syllabification. The present analysis for syllabification follows maximum onset principle (Selkirk, 1984; Selkirk, 1981). The maximum onset principle is a sufficiency condition as demonstrated by the following examples.

i. मृत्युंजय (A name) → [mri] [tjun] [dʒəj]
→ *[mrit][jun][dʒəj]
→ *[mritj][un][dʒəj]
→ *[mrit][jundʒ][əj]

ii. कबूतर (Pigeon) → [kə] [bu] [tər]
→ *[kəb] [ut] [ər]
→ *[kə] [but] [ər]

In the above examples, the right hand side shows the potential syllable structures for a word. The square bracket denotes the syllable boundary. An asterisk before the syllable structure indicates that this potential syllable sequence is incorrect. The above examples show that either onsets are maximized or is equal to number of coda consonants for correct syllabification. It implies that the maximum onset principle hold for syllabification

[2]Hindi graphemes and its corresponding phoneme using international phonetic alphabet (IPA) and Roman symbols are described in Table 10.

in Hindi. Based on many such examples, following regular expressions are proposed for syllabification in Hindi.

i. v → [v]
ii. vv → [v][v]
iii. c*vcv → [c* v] [cv]
iv. c*vc1cv → [c*vc1] [cv]
v. c*vc1c2v → [c*v] [c1c2v]
vi. c*vc1c1v → [c*vc1] [c1v]
vii. c*vc2c2v → [c*vc2] [c2v]
viii. c*vcccv → [c*vc] [ccv]

In the above expressions, v denotes a vowel, "c1" denotes a stop consonant, "c2" represents a semivowel (r, l, v, j) and "c" at intervocalic position denotes a consonant that is neither a stop nor a semivowel but can be any consonant at non-intervocalic position. An asterisk denotes the kleene star.

The finite state machine for the I-level syllabification is shown in Figure 2. It contains seventeen states with the start state as I and the final state as F. The orthography of an input word is transliterated into a sequence of consonants and vowels. The 8 syllabification rules are applied to this sequence to derive a symbol sequence in terms of c, c1, c2 and v. The symbol sequence is the input to the FSM in Figure 1. An arc between a pair of states in FSM is associated with a label (a pair of symbols seperated by "/"). Suppose the symbol pair associated with an arc is "x/y". This indicates that whenever a symbol "x" is fed to the state at the beginning of the arc, the system makes a transition along the arc and outputs the symbol "y". The label e/e symbolizes null input and null output for a transition. If part of a symbol string reaches to the final state F, then it is consumed, and a transition from F to I with arc label e/b takes place, where e is null and b denotes the syllable boundary of the consumed string. The remaining part of the string repeats the same process from initial state I until everything is consumed.

3.1 Syllable Labeling

Hindi is a quantity sensitive language. Therefore, syllables in Hindi are also described based on an attribute called syllable weight or moraic weight (Hayes, 1980; Pandey, 1989). A phonetic, phonological and typological description of syllable weight can be found in (Gordon, 2007). The following rules are used for label syllables based on syllable weights (for examples, see Table 1).

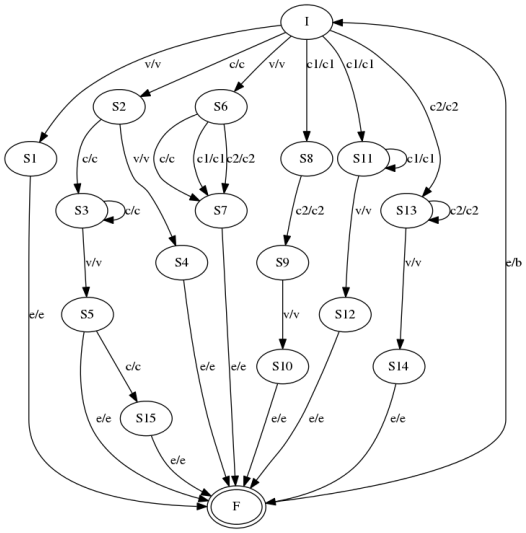


Figure 2: Finite State Machine for Syllabification

i. Each short vowel (like ə,u,i) and each coda consonant of a syllable are assigned a weight of one mora, while a long vowel (like aː,uː,iː) is assigned a weight of two moras.

ii. The syllables with one, two and three moras are called weak (w), heavy (h) and superheavy (sh) syllables respectively (Pandey, 1989; Hayes, 1989; Pandey, 1990).

Sylable Weight	Syllable Label	Gloss
1	[ki]^{w}	that
2	[mən]^{h}	soul
3	[gaːl]^{sh}	cheek

Table 1: Syllable labels according to syllable Weight

A finite state machine for syllable labeling is shown in Figure 3. The machine consists of one initial state (state I), seven non-final states and three final states (F1, F2 and F3). The syllabified string (i.e, the syllable boundary marked as b) is given as input to the initial state. Since, each short vowel gets one mora and long vowels get two moras, therefore, vowel type distinction is essential at syllable labeling stage. Each coda consonants get one mora and the onset consonants do not contribute to the moraic weight of syllables in Hindi. Therefore, consonant type distinction is not required at this stage. The symbol c, v_s and v_l stand for consonants, short vowels and long vowels respectively. The output symbol along an arc is either a syllable label or the reflection of the input itself. There is a null transition from each fi-

nal state to the initial state so that the process can be repeated for the remaining part of the string. The FSM assigns the label "w" to a syllable with phoneme sequences v_s, cv_s, ccv_s, ccc*v_s and "h" to v_sc , c*v_sc, c*v_l, and "sh" to c*v_scc, c*v_lc.

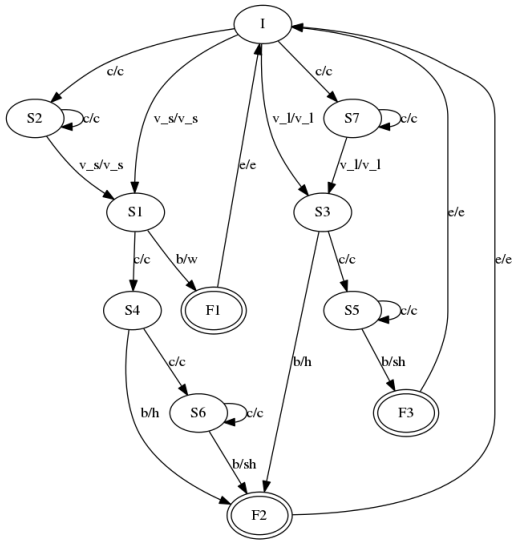


Figure 3: Finite State Machine for Syllable Labeling

4 Foot Formation

The concept of foot or feet is brought to linguistic from poetry. The notion of foot work as a metric to define stress pattern in a language (Jakobson, 1960). Moreover, foot also plays an important role in resyllabification due to deletion of a segment (Selkirk, 1996). In this section, a new approach of foot formation is described for Hindi. The approach is based on three assumptions and six rules. The rules apply in the direction from left to right.

4.1 Assumptions

i. Foot is formed using the labeled syllables of a word. The process of syllable labeling is described in the section 3.

ii. Foot is either binary branching or projected on at least a bimoraic syllable.

iii. A superfoot is formed either between a syllable and a foot, or between two foot.

4.2 Rules

The six rules for foot formation are listed below.

i. Weak to Weak Affinity Rule (WWAR): Two adjacent weak syllables form a binary foot and results into a bimoraic foot as shown in the Figure 4.

The foot $<\sum_s>$is the extra metrical foot, which never bears any stress.

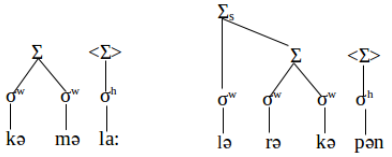


Figure 4: Bimoraic binary foot

ii. Heavy to Weak Affinity Rule (HWAR): A heavy and a weak adjacent syllables form trimoraic binary foot as shown in the Figure 5.

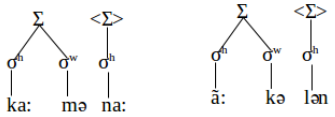


Figure 5: Trimoraic foot

iii. Weak to Heavy Affinity Rule (WHAR): This kind of foot is either formed in bisyllabic Hindi words or in loan words as shown in the Figure 6.

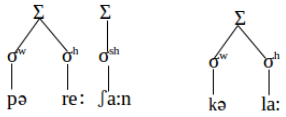


Figure 6: Trimoraic foot

iv. Heavy to Heavy Affinity Rule (HHAR): Two adjacent heavy syllables also form a binary foot as shown in the Figure 7.

v. Superheavy to Others Affinity Rule (SOAR): The superheavy syllables always projected as a foot as shown in Figure 7. The superfoot ($\sum_s$) is formed using one syllable and one foot. The projected foot constituent in $\sum_s$ could be either a new syllable after schwa deletion as shown in Figure 4 or a syllable which inherently bear stress i.e., the superheavy syllable as shown in Figure 8.

vi. List Affinity Rule (LAR): LAR is devised for handling words having same syllable structure at underlying phonemic form but realized differently at surface level. Such overlapping cases are stored in different list (usually different spreadsheets) and different foot formation rules are applied to these lists. The foot formation rules shown in Figure 9 describe four cases with overlapping syllable structure. These four cases represent four different list of words. These rules are called

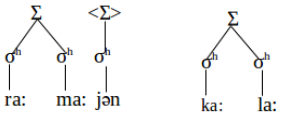


Figure 7: Heavy syllables forming a foot

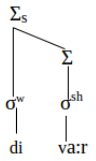


Figure 8: Superheavy syllables forming foot

as affinity hierarchy rules. The word affinity describes the interaction between different or similar type of syllables. The word 'hierarchy' is used because these rules apply in the order of their height. The top rule in the hierarchy applies first and so on. The decreasing order of height of these rules are LAR>WWAR >HWAR >WHAR >HHAR >SOAR.

5 Schwa Deletion and Resyllabification

Schwa deletion is an optional phenomena in Hindi. This means that schwa can be deleted or retained in the same environment, and the choice solely depends on the speaker and the context being used. Schwa deletion phenomena in Hindi helps speakers to utter a word quickly, i.e., the process of schwa deletion reduces the overall effort in terms of duration. It enables stress shift from one syllable to other. The following rules describe the contexts in which schwa gets deleted.

i. $\text{ə} \rightarrow \Phi / [\sigma^{\text{w}} - \sigma^{\text{w}}]_{\sum}$

ii. $\text{ə} \rightarrow \Phi / [\sigma^{\text{h}} - \sigma^{\text{w}}]_{\sum}$

iii. $\text{ə} \rightarrow \Phi / [\sigma^{\text{sh}} - \sigma^{\text{w}}]_{\sum}$

In other words, if the right most node of a foot is a weak syllable having schwa then schwa can be deleted.Application of the above schwa deletion (SD) rules and consequent re-syllabification of exemplar words are shown in Table 3. The process of schwa deletion and re-syllabification occur at foot level. In the process of re-syllabification, the bare consonant(s) after schwa deletion are assigned as coda consonant(s) to the preceding syllable. The foot structure for the examples in Table 2 can be found in the Section 4.

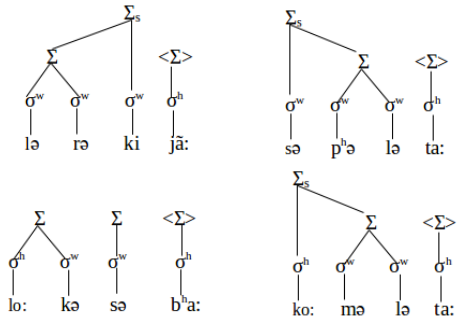


Figure 9: Examples of words with same syllable structure but different foot formation

UPF	SD	Resyllab	Gloss
kəməla:	kəmla:	[kəm][la:]	A Name
lərəkəpən	lərəkpən	[lə][rək][pən]	Childhood
ka:məna:	ka:mna:	[ka:m] [na:]	Wish
lo:kəsəb^{h}a:	lo:ksəb^{h}a:	[lo:k][sə][b^{h}a:]	Parliament
səp^{h}ələta:	səp^{h}əlta:	[sə] [p^{h}əl] [ta:]	Success

Table 2: Examples of word re-syllabification (Resyllab) after applying schwa deletion (SD) to underlying phonemic form (UPF)

6 Anusvara and Anunasika Pronunciation

Over the past, researchers have ignored the ambiguities in the pronunciation of anusvara/anunasika for G2P conversion. A simple finite state transducer for anusvara and anunasika is proposed by (Choudhury, 2003). (Pandey, 2014) says that the pronunciation ambiguities in anusvara/anunasika can be solved by preparing an exhaustive list of irregular cases. He further advocates the need for revision in the orthography to get rid of these irregular cases. However, not only in superscripted vowel diacritics as described in (Pandey, 2014), but in general, the present day Devanagari uses bindu and chandrabindu interchangeably. Some such examples are shown below in Table 3. The process of mapping anusvara and anunasika to appropriate phoneme is based on the observations and rules described below. The proposed approach maps both bindu and chandrabindu to the same phoneme X at the underlying phonological level. A single phoneme X for both anusvara and anunasika at underlying phonological level correctly captures the phonological structure of a word. The use of single phoneme X transforms the co-domain of the function G2P:A→ B and it becomes a one-to-one function from many-to-one

function as shown in Fig. 10 and Fig.11 . The disambiguation rules apply on the phoneme X. Hence, these rules perform better for the words with identical use of bindu and chandrabindu. Moreover,The proposed approach uses both supra-segmental and segmental phonological constraints for anusvara/anunasika disambiguation.

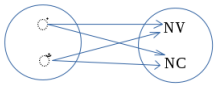


Figure 10: One-to-many mapping due to the identical use of anusvara and anunasika

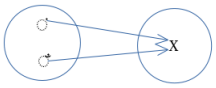


Figure 11: The use of phoneme X transform the codomain of G2P function. It becomes one-to-one function.

6.1 Observations

- Nasalization of vowel (NV) does not change the moraic weight of a syllable while the homo-organic nasal (HN) increases the moraic weight of a syllable by one unit.
- A syllable having phoneme for either anusvara or anunasika always tries to keep minimum moraic weigth difference with its succeeding syllable.

6.2 Rules

- Initially both anusvara and anunasika is mapped to phoneme, say, X. The moraic weight of X is assumed as one unit.
- The decision of using NV or HN is based on the comparison of moraic weight between the syllable to be mapped for NV/HN and the syllable succeeding to it.
  - If the moraic weight of the syllable containing phoneme X is greater than that of the next syllable, then replace the vowel and phoneme X by the corresponding NV.
  - If the moraic weight of the syllable containing phoneme X is less than or equal to that of the next syllable, then replace the phoneme X by the HN corresponding to the following phoneme.
- If the word final syllable contains the phoneme X at the last coda position then nasalize the vowel preceding X and delete X.
- If the word final syllable contains the phoneme X at non-final position and followed by a tʃ, tʃ^{h}, dʒ and dʒ^{h} then nasalize the vowel preceding X and delete X. Otherwise replace X by HN corresponding to the following phoneme.

Table 4 and 5 show examples of application of the above written rules for phonemic realization of anusvara and anunasika. In Table 4, the acronym Syllab denotes the syllable division and Mw denotes the moraic weight of syllables in the ordered pair.

7 Data Driven G2P Systems for Hindi

Two data-driven G2P models are trained on an expert annotated training lexicon of size 26454 words. These words are extracted from BBC Hindi. The first model is a joint sequence (JS) based G2P model trained using the sequitur (Bisani and Ney, 2008) toolkit. The second G2P model is a bidirectional deep LSTM model. The model configuration is same as reported in (Rao et al., 2015). Three forward and three backward hidden layers with 256 nodes at each layer is used. The output layer is a connectionist temporal classification (CTC) (Graves et al., 2006) layer and the error function is softmax.

8 Results

The publicly accessible Hindi wordnet (Bhattacharyya, 2010) is used for the testing purpose. The wordnet is first cleaned i.e., digits, hyphen and other special characters are removed. Long words especially compounds, derived and inflected words are picked up from different lexical categories like Noun, Verb, Adjective and Adverbs. The first list contains 3500 words for which schwa deletion rule applies at least once. A second list 700 words having diacritic for anusvara or anunasika. This implies that two test sets are used containing 3500 and 700 words. These sets are annotated by expert at three levels i) phonemic sequence, ii) syllable boundaries, and iii) stress mark. The G2P output for the

Graphemic Form-1	Graphemic Form-2	Correct Surface Form	Gloss
ऊँट	ऊंट	ũːʈ	Camel
कारवाँ	कारवां	kaːrvãː	coffle
कुँवारी	कुंवारी	kũvaːriː	Unmarried Girl
गाँधी	गांधी	gaːndhiː	A Name
गेहूँ	गेहूं	geːhũː	Wheat
घुँघरू	घुंघरू	gʰuŋgʰ ruː	A Name
जाँघिया	जांघिया	dʒaːŋgʰijaː	Underwear
जाएँ	जाएं	dʒaːẽː	Go(Honorific)
ढाँचे	ढांचे	ɖãːtʃeː	Shape
ताँबा	तांबा	taːmbaː	Copper
फँसा	फंसा	pʰə̃saː	Trap (past)
भँवरी	भंवरी	bʰə̃vriː	Loop
वर्षगाँठ	वर्षगांठ	vərʂgãːʈʰ	Anniversary
शाहजहाँ	शाहजहां	ʃaːhdʒəhãː	A Name
हालाँकि	हालांकि	haːlãːkiː	However

Table 3: Examples of word in which diacritic mark for ansuvara and anunasika are used interchangeably at orthographic level but only one phoneme (i.e., either nasalized vowel or nasal consonant) emerges at surface phonemic form level

Grapheme	Syllab	Mw	Decision	Gloss
अंगूर	[aX] [guːr]	(2,3)	X=HN=ŋ	Grape
चींटी	[tʃiːX][ʈiː]	(3,2)	iːX=NV=ĩː	Ant
अंबर	[əX][bər]	(2,2)	X=HN=m	Sky
अंधा	[əX][ɖʰaː]	(2,2)	X=HN=n	Blind
आँचल	[aːX][tʃəl]	(3,2)	aːX=NV=ãː	A Name
सिंचाई	[siX][tʃaːiː]	(2,2)	X=HN=n	irrigation
ज़ंजीर	[dʒəX][dʒiːr]	(2,3)	X=HN=n	chain
अंधेरे	[əX][dʰeː][reː]	(2,2)	X=HN=n	Dark
आंवले	[aːXv][leː]	(3,2)	aːX=NV=ãː	gooseberry

Table 4: Examples of applications of rules for phonemic realization of anusvara and anunasika

Grapheme	Phoneme	Gloss
गमलों	gəmlõː	Flowerpots
अनंत	ənənt	Infinity
धीरेंद्र	dʰiːreːndr	A name
पेंच	pẽːtʃ	Bolt
पाँच	pãːtʃ	Five

Table 5: Examples of applications of rule (Rule iii and iv) for phonemic realization of anusvara and anunasika

test sets by the proposed system, the data driven system and the previous systems are compared against the annotated test test. The rules of previous systems (Narasimhan et al., 2004),(Choudhury, 2003), (Bali et al., 2004) and (Pandey, 2014) are implemented in Python for comparison on the same test set. The example words where the others failed and current system succeded is shown in Table 9.The performance of these systems is reported below in Table 7. The present work has one limitation though.

It cannot predict the stress pattern for the words having different part of speech categories as described in (Pandey, 2014) and (Dyrud, 2001). However, the number of such words in Hindi is small and can be listed. The future work will include a mechanism to predict stress for these words based on their part-of-speech category.

9 Prediction of Stressed Syllables

The process of resyllabification discussed in section 5 can upgrade syllables from weak to heavy or heavy to superheavy. Therefore, the after resyllabification the syllabified strings are fed to the FSM for syllable labeling. The labels assigned by FSM after resyllabification is called prosodic label. Table 6 shows examples of applications of syllable stress rule. Here a stressed syllable is preceded by a stress mark (ˈ). The following rules translate the prosodic labels into stressed/unstressed component.

Type	Stress Mark	Gloss
w+h	ˈkəlaː	Art
h+h	ˈkaːlaː	Black
sh+h	aːˈraːm	Comfort
sh+sh	ˈraːmˈnaːtʰ	A name
w+h+h	məˈhiːnaː	Month
sh+h+h	ˈaːlˈmaːriː	cupboard
h+h+sh	ˈhindusˈtaːn	Country Name

Table 6: Examples of applications of syllable stress rule

Systems	%Error1	% Error2
narasimhan et. al	9.57	12.08
choudhury	6.28	17.6
bali et. al.	5.2	8.27
pandey	7.5	9.4
JS Model	2.6	7.37
LSTM Model	2.16	3.56
Current System	0.45	1.5

Table 7: A summary of comparison of performance of the current system with previous rule-based systems and the state of the art data driven systems. The %Error1 and %Error2 denotes the word error rate due to schwa deletion and anusvara/anunasika pronunciation respectively.

i. Superheavy syllables are always stressed.

ii. Heavy syllables at the ultimate position are unstressed and stressed otherwise.

iii. Weak syllables at the penultimate position in bisyllabic words are stressed and unstressed otherwise.

The output of current system is evaluated for syllabification and stress prediction for the test set described in Section 8. The report is summarized in Table 8.

Testing Level	% Accuracy
Syllabification	100
Stressed Syllables	99.34

Table 8: A summary of testing of the current implementation for syllabification and prediction of stressed syllables

10 Conclusion

In this paper, a new approach for deriving Hindi word prosody is described. The WPC uses a novel rule-based G2P converter which outperforms the state of the art data-driven G2P systems and the previous rule-based system for Hindi. The proposed G2P system uses non-linear phonological rules with the provision of recursive foot. The proposed system has one limitation though. The system cannot predict the correct stress pattern of a word having two part-of-speech category as described in (Pandey, 2014). The proposed work can be further utilized in prosodic analysis by extracting stressed/unstressed syllables at textual level. The acoustic analysis can be performed by training the speech data and corresponding text using hidden markov model (HMM) or deep neural networks (DNN).

Grapheme	Other Systems	Current System	Gloss
लोकसभा	loːkəsbʰaː	loːksəbʰaː	Parliament
ताजमहल	taːdʒmhəl	taːdʒməhəl	A Name
कमलनयन	kəmlnəjən	kəməlnəjən	A Name
अनुसरण	anusrəɳ	anusərəɳ	To follow
अपवचन	əpəʋcən	əpʋəcən	Bad words
अपशकुन	əpəʃkun	əpəʃkun	Bad omen
बहुवचन	bəhuʋcən	bəhuʋəcən	Plural
उपग्रह	upəgrəh	upgrəh	satellite
हरभजन	hərəbʰdʒən	hərbʰədʒən	A name
आंकने	aːŋkneː	ãːkneː	To Judge
क्योंकि	kjoːnkiː	kjõːkiː	Because
टाँग	ʈãːg	ʈaːŋg	Leg
पसलियां	pəsəlijãː	pəslijãː	Ribs

Table 9: Examples words where the others failed and current system succeeded

Acknowledgement

I would like to thank Pramod Pandey and Samudravijaya K for their comments and suggestions on the initial draft of this paper. I would also like to thank all the reviewers for their valuable comments.

References

Diana Apoussidou. 2006. *The learnability of metrical phonology*. Netherlands Graduate School of Linguistics.

Kalika Bali, Partha Pratim Talukdar, N Sridhar Krishna, and AG Ramakrishnan. 2004. Tools for the development of a hindi speech synthesis system. In *Fifth ISCA Workshop on Speech Synthesis*.

Devanagari	अ	आ	इ	ई	उ	ऊ	ए	ऐ	ओ	औ	क	ख	ग	घ
IPA	ə	aː	i	iː	u	uː	eː	æː	oː	əoː	k	k^h	g	g^h
Roman	a	aa	i	ii	u	oo	e	ei	o	au	k	kh	g	gh
Devanagari	च	छ	ज	झ	त	थ	द	ध	न	ट	ठ	ड	ढ़	ण
IPA	tʃ	$tʃ^h$	dʒ	$dʒ^h$	t	t^h	d	d^h	n	ʈ	$ʈ^h$	ɖ	$ʈ^h$	ɳ
Roman	c	ch	j	jh	t	th	d	dh	n	tx	txh	dx	dxh	nx
Devanagari	प	फ	ब	भ	म	य	र	ल	व	श	ष	स	ह	क्ष
IPA	p	p^h	b	b^h	m	j	r	l	ʋ	ʃ	ʂ	s	h	kʂ
Roman	p	ph	b	jh	m	y	r	l	v	sh	sx	s	h	ksh
Devanagari	त्र	ज्ञ	क़	ड़	फ़	ङ	◌ँ	◌ं	◌ः	◌्	◌़	◌ि	◌ी	matra उ ऊ
IPA	tr	z	q	ɽ	f	ŋ	eː	DMC	DMB	æː	RH	i	iː	u uː
Roman	tr	z	q	rx	f	ng	e	NV/NC	NV/NC	ei	r	i	ii	u oo

Table 10: Hindi grapheme and its corresponding phoneme in IPA and Roman. The DMC and DMB means Diacritic mark chandrabindu and bindu respectively. RH represents the Ra halant which is also a diacritic mark. u and oo in the last column represent matra (diacritic mark) for the vowel उ and ऊ respectively.

Ashwin Bellur, K Badri Narayan, K Raghava Krishnan, and Hema A Murthy. 2011. Prosody modeling for syllable-based concatenative speech synthesis of hindi and tamil. In *Communications (NCC), 2011 National Conference on*, pages 1–5. IEEE.

Barbara Bernhardt and John Gilbert. 1992. Applying linguistic theory to speech–language pathology: the case for nonlinear phonology. *Clinical Linguistics & Phonetics*, 6(1-2):123–145.

Pushpak Bhattacharyya. 2010. Indowordnet. In *In Proc. of LREC-10*. Citeseer.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication*, 50(5):434–451.

Alan W Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules.

Monojit Choudhury. 2003. Rule-based grapheme to phoneme mapping for hindi speech synthesis. In *90th Indian Science Congress of the International Speech Communication Association (ISCA), Bangalore, India*.

Megan J Crowhurst. 1994. Foot extrametricality and template mapping in cupeño. *Natural Language & Linguistic Theory*, 12(2):177–201.

Lars O Dyrud. 2001. *Hindi-Urdu: Stress accent or non-stress accent?* Ph.D. thesis, University of North Dakota.

Aravind Ganapathiraju, Jonathan Hamaker, Joseph Picone, Mark Ordowski, and George R Doddington. 2001. Syllable-based large vocabulary continuous speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 9(4):358–366.

Matthew Gordon. 2007. *Syllable weight: phonetics, phonology, typology*. Routledge.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM.

T Alan Hall and Ursula Kleinhenz. 1999. *Studies on the phonological word*, volume 174. John Benjamins Publishing.

Bruce Philip Hayes. 1980. *A metrical theory of stress rules*. Ph.D. thesis, Massachusetts Institute of Technology.

Bruce Hayes. 1989. Compensatory lengthening in moraic phonology. *Linguistic inquiry*, 20(2):253–306.

Bruce Hayes. 1995. *Metrical stress theory: Principles and case studies*. University of Chicago Press.

Roman Jakobson. 1960. Linguistics and poetics. In *Style in language*, pages 350–377. MA: MIT Press.

S Prahallad Kishore and Alan W Black. 2003. Unit size in unit selection speech synthesis. In *INTERSPEECH*.

A Lakshmi and Hema A Murthy. 2006. A syllable based continuous speech recognizer for tamil. In *INTERSPEECH*.

Mark Liberman and Alan Prince. 1977. On stress and linguistic rhythm. *Linguistic inquiry*, 8(2):249–336.

John J McCarthy and Alan S Prince. 1990. Foot and word in prosodic morphology: The arabic broken plural. *Natural Language & Linguistic Theory*, 8(2):209–283.

Bhuvana Narasimhan, Richard Sproat, and George Kiraz. 2004. Schwa-deletion in hindi text-to-speech

synthesis. *International Journal of Speech Technology*, 7(4):319–333.

Josef R Novak, Nobuaki Minematsu, and Keikichi Hirose. 2012. Wfst-based grapheme-to-phoneme conversion: Open source tools for alignment, model-building and decoding. In *FSMNLP*, pages 45–49.

Manjari Ohala. 1983. *Aspects of Hindi phonology*, volume 2. Motilal Banarsidass Publisher.

Vincent Pagel, Kevin Lenzo, and Alan Black. 1998. Letter to sound rules for accented lexicon compression. *arXiv preprint cmp-lg/9808010*.

Pramod Kumar Pandey. 1989. Word accentuation in hindi. *Lingua*, 77(1):37–73.

Pramod Kumar Pandey. 1990. Hindi schwa deletion. *Lingua*, 82(4):277–311.

Pramod Pandey. 2014. Akshara-to-sound rules for hindi. *Writing Systems Research*, 6(1):54–72.

Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays. 2015. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4225–4229. IEEE.

Elisabeth O Selkirk. 1980. The role of prosodic categories in english word stress. *Linguistic inquiry*, 11(3):563–605.

Elisabeth O Selkirk. 1981. English compounding and the theory of word structure. *The scope of lexical rules*, pages 229–277.

Elisabeth O Selkirk. 1984. On the major class features and syllable theory.

Elisabeth O Selkirk. 1986. *Phonology and syntax: the relationship between sound and structure*. MIT press.

Elisabeth Selkirk. 1996. The prosodic structure of function words. *Signal to syntax: Bootstrapping from speech to grammar in early acquisition*, 187:214.

Paul Taylor. 2005. Hidden markov models for grapheme to phoneme conversion. In *Interspeech*, pages 1973–1976.

Dong Wang and Simon King. 2011. Letter-to-sound pronunciation prediction using conditional random fields. *IEEE Signal Processing Letters*, 18(2):122–125.

Three-phase training to address data sparsity in Neural Machine Translation

Ruchit Agrawal
LTRC
IIIT Hyderabad

Mihir Shekhar
DSAC
IIIT Hyderabad

Dipti Misra Sharma
LTRC
IIIT Hyderabad

Abstract

Data sparsity is a key problem in contemporary neural machine translation (NMT) techniques, especially for resource-scarce language pairs. NMT models when coupled with large, high quality parallel corpora provide promising results and are an emerging alternative to phrase-based Statistical Machine Translation (SMT) systems. A solution to overcome data sparsity can facilitate leveraging of NMT models across language pairs, thereby providing high quality translations despite the lack of large parallel corpora. In this paper, we demonstrate a three-phase integrated approach which combines weakly supervised and semi-supervised learning with NMT techniques to build a robust model using a limited amount of parallel data. We conduct experiments for five language pairs (thereby generating ten systems) and our results show a substantial increase in translation quality over a baseline NMT model trained only on parallel data.

1 Introduction

Neural Machine Translation (NMT) is an emerging technique which utilizes deep neural networks (Kalchbrenner and Blunsom, 2013), (Sutskever et al., 2014), (Bahdanau et al., 2014) to generate end-to-end translation. NMT has shown promising results for various language pairs and has been consistently performing better than Phrase based SMT, the state-of-the-art MT paradigm until a few years back. A major benefit in NMT which makes it so popular is its ability to use deep neural networks and learn linguistic information from the parallel data itself without being fed any learning features. This makes it a conceptually simple method which provides significantly better translations than other MT paradigms like rule-based MT and statistical MT. Furthermore, it eliminates the need for complex feature engineering by providing end-to-end translation. The newly proposed attention mechanism is a valuable addition to NMT contributing to significant gain in performance.

NMT systems have achieved competitive accuracy scores under large-data training conditions for language pairs such as En $\rightarrow$ Fr (English - French) and En $\rightarrow$ De (English - German). However, on the other hand, NMT models are unable to extract sufficient linguistic information in terms of morphology, word order, syntactic structure and semantics in low resource scenario. This makes translation among morphologically rich languages especially challenging. Also, due to the unavailability of large parallel corpora, the vocabulary size tends to be low, due to which any word which is not included in the vocabulary is mapped to a special token representing an unknown word $[UNK]$. This causes a large number of $[UNK]$'s in the target sentence, which results in a drastic drop in the translation quality. This behaviour makes vanilla NMT a poor choice for low resource language pairs, especially if they are morphologically rich.

In this paper, we propose an integrated approach for reducing the impact of data sparsity in NMT, which leverages a large monolingual corpus of the source language, which is easier to obtain in comparison to parallel corpus. We employ a small parallel corpus in addition to the monolingual

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 13–22

corpus, and through a combination of weakly-supervised and semi-supervised learning, we build an efficient model which delivers promising results. Our approach along with the intuition driving it is described in detail in Section 4. Our model obtains an improvement of five to eight points in BLEU score over an attention based encoder-decoder model trained over a parallel corpus. The results obtained on test sets from different domains are also promising, which suggests that the proposed model is able to perform domain adaptation successfully due to the presence of a rich vocabulary learnt from hree-phase training.

The main contributions of our work are :

- We propose an integrated approach which combines weakly-supervised learning and semi-supervised learning to reduce the impact of data sparsity on NMT, by utilizing a large monolingual corpus of the source language in addition to a small parallel corpus.
- We tweak the NMT architecture to generate optimum performance and conduct experiments on different Indian language pairs using the proposed approach. We demonstrate that we are able to build a robust NMT model which produces quality translation and delivers promising results, significantly better than a baseline NMT model.

2 Related Work

NMT methods are data hungry. Efficient NMT for Indian languages is a challenging problem, owing to multiple reasons including morphological complexity and diversity, in addition to a lack of resources for many languages. Advances in the recent past mainly employ statistical and rule based methods for MT. (Kunchukuttan et al., 2014) uses statistical phrase based machine translation for Indian Languages using Moses (Koehn et al., 2007) for phrase extraction as well as lexicalized reordering. Sampark (Anthes, 2010) is a transfer based system for translation between 18 Indian language pairs, which uses a common lexical transfer engine, whereas minimum structural transfer is required between Indian languages. (Kunchukuttan and Bhattacharyya, 2016) use orthographic features along with SMT to reach state of the art results in SMT for related languages.
The use of monolingual data to improve translation accuracy in NMT was first proposed by (Gulcehre et al., 2015). Monolingual models were trained independently and then were integrated to decoder module either through rescoring of the beam (shallow fusion), or by adding the recurrent hidden state of the language model to the decoder state of the encoder-decoder network, with an additional controller mechanism that controls the magnitude of the LM signal (deep fusion).
(Sennrich et al., 2016) proposed use of synthetic data, a parallel data corpus generated using back-translation along with parallel corpus to increase the translation accuracy.

Our method differs from them since it is three-phased. In the first phase, we train our model over a synthetic corpus generated using a suboptimal MT technique, and then fine tune it further on gold data. This allows better control over training during various stages - leading to better translation quality for Indian languages. Our second phase is inspired from (Zoph et al., 2016). They use transfer learning to increase translation quality between resource scarce language pairs by incorporating the weights learnt during training for high resource language pairs. It was also found that languages having similar structure, like Fr $\longleftrightarrow$ En (French - English) showed better improvement in performance as compared to other languages having little similarity, like Uz $\longleftrightarrow$ En (Uzbek - English). Our approach is based on the intuition that transfer learning between the same language pair should perform better than its multilingual counterpart. The experimental results described in Section 5 demonstrate that the above intuition stands correct. During fine-tuning, the change in weights in each epoch learnt through transfer learning allows the model to align more towards the correct model.
(McClosky et al., 2006) proposed using self-training for the task of parsing. We have experimented with its use in Neural machine

translation.

3 Experimental Setup

3.1 Datasets

We employ a small parallel corpus and large monolingual corpora for training. For the former, we use the multilingual Indian Language Corpora Initiative (ILCI) corpus [1], which contains 50,000 sentences from the health and tourism domains aligned across eleven Indian languages. We employed manual preprocessing to eliminate misalignments - the resultant dataset has a size of 47,382 sentences. These are split randomly into training set, validation set and test set containing 44,000, 1382 and 2000 sentences respectively.

Table 1: Corpus statistics - ILCI

	Tokens	Vocabulary
hin	850968	39170
pan	849679	849679
guj	759380	62780
tam	849679	86462
ben	715886	50553
urd	832776	36738
tel	632995	86997
kon	643605	70030
eng	808370	35134
mar	663597	77057
mal	599422	101869

The statistics for the ILCI corpus are given in Table 1. We use the EMILLE monolingual corpora (McEnery et al., 2000) for five languages and the UrMonoCorp (Jawaid et al., 2014) for Coarse Learning detailed in Section 4.[2]. These statistics are given in Table 2. In addition to these, we extract samples from the EMILLE (McEnery et al., 2000) parallel corpus for the Housing and Legal domains. These datasets are used as test sets to show coverage of our NMT model. Details are given in Table 3.

[1]This corpus is available on request from TDIL : https://goo.gl/VHYST

[2]We extract a sample containing 500,000 sentences from UrMonoCorp

Table 2: Monolingual Corpora statistics - EMILLE and *UrMonoCorp

	Sentences	Tokens	Vocabulary
hin	612705	11986152	321356
pan	488985	14285063	272771
tam	827439	17170697	1285031
guj	272526	12766111	660465
ben	259145	2671369	243531
urd*	500000	8744825	157133

Table 3: Parallel Corpus Statistics - EMILLE. H: Housing, L: Legal

		Sentences	Tokens	Vocabulary
hin	H	1183	23178	3131
	L	1321	27700	3880
ben	H	1109	17815	3310
	L	1288	21690	4567
guj	H	1113	17537	4405
	L	1382	21377	5689
pan	H	1308	20729	3771
	L	1368	24971	3763
urd	H	1327	22691	2871
	L	1386	27207	3945

3.2 Resources

For our experiments, we use synthetic data in addition to the gold data (described in detail in Section 4) to compensate for the relatively lower size of our gold corpus. The generation of synthetic data from the monolingual corpora is done using the *Sampark* (Anthes, 2010) systems, which are available for 9 Indian language pairs[3]. Sampark is a multipart machine translation system developed under the Indian Language Machine Translation project. It uses a transfer-based engine and has a huge repository of rules for dealing with Indian language specific constructs. The motivation behind this choice for synthetic data generation stems from the quality of performance obtained using Sampark for Coarse Learning due to its uniform coverage and large vocabulary.

3.3 NMT Architecture

The main component of our NMT model is a single neural network trained jointly to provide end-to-end translation. Our architecture consists of two components called encoder and

[3]https://goo.gl/yu7KUT

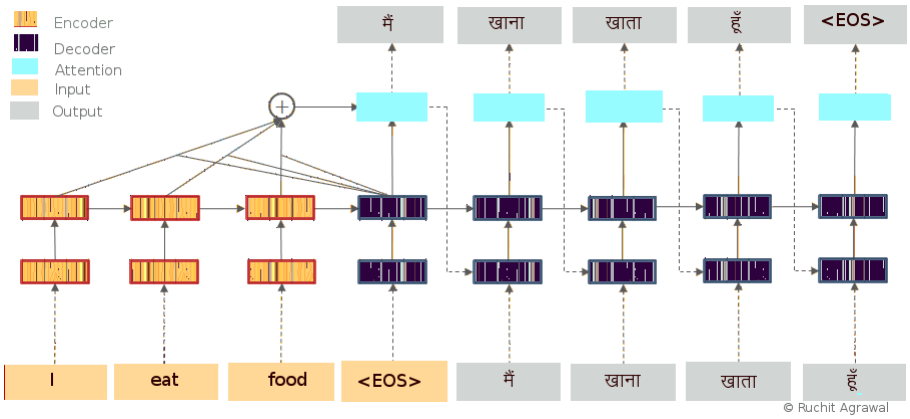


Figure 1: *A simple two-layered encoder-decoder based NMT architecture as proposed by (Sutskever et al., 2014), which translates a source sentence* "I eat food" *into a target sentence* "मैं खाना खाता हूँ" . *'EOS' denotes the end of the sentence.*

decoder, as shown in Figure 1. The components are composed of Stacked RNNs (Recurrent Neural Networks), using either Long Short Term Memory (LSTM) (Sundermeyer et al., 2012) or Gated Recurrent Units (Chung et al., 2015). The encoder encodes the source sentence into a vector from which the decoder extracts the target translation sentences. This facilitates learning of long-distance dependencies, thereby enabling the system to learn an end-to-end model.

Specifically, NMT aims to model the conditional probability $p(y|x)$ of translating a source sentence x $= x_1, x_2...x_u$ to a target sentence y $= y_1, y_2, ...y_v$. Let s be the representation of the source sentence as computed by the encoder. Based on the source representation, the decoder produces a translation, one target word at a time and decomposes the conditional probability as :

$$\log p(y|x) = \sum_{j=1}^{v} \log p(y_j|y_{<j}, x, s) \qquad (1)$$

The entire model is jointly trained to maximize the (conditional) log-likelihood of the parallel training corpus:

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(y^{(n)}|x^{(n)}) \qquad (2)$$

where $(y^{(n)}, x^{(n)})$ represents the n^{th} sentence in parallel corpus of size N and θ denotes the set of all tunable parameters.

(Bahdanau et al., 2014) proposed an attention mechanism so that the memory of the source hidden states is tracked and reference is done to the relevant ones when needed. This increases the translation quality for longer sentences. Further, local and global attention mechanism was proposed by (Luong et al., 2015). We employ encoder-decoder system with LSTM units trained to optimize maximum-likelihood (via a softmax layer) with back-propagation through time (Werbos, 1990). We also use an attention mechanism that allows the target decoder to look back at the source encoder, specifically the local attention plus feed-input model (Luong et al., 2015). We use OpenNMT-Lua (Klein et al., 2017) for building the models. The learning rate is set to 1 for both coarse learning and fine tuning. Our primary motive for the coarse learning stage is to learn only the general features from the synthetic corpus, thereby making it easier to fine tune the model. Hence, the decay rate is set to be 0.9 and 0.97 for coarse learning and fine tuning respectively, which results in significantly faster convergence for the former. Due to the same reason, the dropout ratio is kept higher for coarse learning (0.5) as compared to fine tuning (0.3). As the decay rate is higher for coarse learning, we run it for considerably lower number of epochs (40 epochs) as compared to fine tuning (130 epochs).

We performed grid search to obtain best set of hyper-parameter with validation data for each phase including learning rate, learning decay-rate and drop out. We did hyper-parameter tuning for Hindi-Gujarati language

pairs and used the same parameters values for corresponding to each phases for all the other language pairs. Some of the parameters like optimisation function, word vector size and brnn parameters were set to the default values. Detailed set of parameters used is provided in Table 4.

4 Experiments using Three-Phase Training

We train a baseline NMT model using the small parallel corpus (ILCI) described in Section 3.2. We call this model NMT_{Base}. In order to compare our results with the state-of-the-art, we train a phrase based SMT model using the same corpus. The SMT model is trained using Moses (Koehn et al., 2007) for phrase extraction and lexicalized reordering as described in (Kunchukuttan et al., 2014)[4]. We call this model SMT_{SA}. In this section, we describe a three-phase integrated approach which leverages a large monolingual corpus of the source language and an existing MT tool to improve translation accuracy as well as domain coverage.

Figure 2 shows the block diagram of this approach. The entire process is divided into three stages : **Coarse learning**, **Fine-tuning** and **Self-training**. We begin by Coarse Learning, which can be thought of as providing the neural model with some information about grammatical constructs of the target language. The second phase employs Fine-tuning to enrich the linguistic knowledge of the model with the help of a hand-annotated gold parallel corpus. This is then followed by self-training, where the fine-tuned model is employed to generate a synthetic corpus again, on which we perform Coarse Learning for the next training iteration. Thus, this is a cyclical process, which is stopped when further increase in accuracy is observed to be negligible.
The following sections explain the three phases in detail :

[4]We train our own SMT model since the training, validation and testing sets used by Sata-Anuvadak are unavailable to us.

4.1 Coarse Learning

Coarse Learning is a form of weak supervision, which is a machine learning paradigm where the model learns from noisy data or prior knowledge. (Haghighi and Klein, 2009) used rich syntactic and semantic features to induce prior knowledge for the task of coreference resolution. (Ratner et al., 2016) uses an ensemble of weak learners using rules to identify biomedical entities from medical documents.

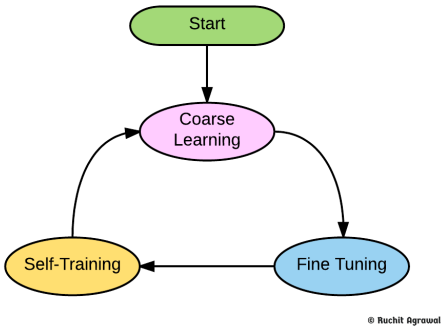


Figure 2: *Three-phase approach to improve robustness and accuracy. The entire cycle is repeated until the increase in accuracy is minimal. We conduct three self-training iterations.*

Large annotated parallel corpora are not easy to obtain for Indian languages. However, it is easier to use an existing MT system to generate a sub-optimal translation of a monolingual corpus, which is referred to as synthetic data.
Building upon this insight, we generate a synthetic corpus for 10 language pairs[5] using *Sampark* (Anthes, 2010) to translate the EMILLE monolingual corpora. We use this tool rather than *Sata − Anuvadak* (Kunchukuttan et al., 2014) due to its uniform domain coverage - a trait desirable for synthetic data generation when dealing with multiple domains

We train an NMT model over the synthetic corpus thus generated. This helps the model to learn significant linguistic information about the target language in the form of syntax, word order and morphology, along with the vocabularies, although with certain

[5]Language pairs for which both large monolingual corpora and *Sampark* were available.

Table 4: Detailed parameters for training the NMT models.
LR : Learning Rate, DS : Start Decay at

Phase	Parameters							
	Sample	WordVecSize	Layers	Dropout	LR	LR Decay	DS	Epochs
Baseline	80%	500	2	0.2	0.76	0.325	10	60
Fine Tuning	80%	500	2	0.3	0.5	0.15	10	60
Coarse ST1	50%	500	2	0.55	0.9	0.75	5	30
Fine ST1	80%	500	2	0.2	0.8	0.25	10	60
Coarse ST2	50%	500	2	0.4	0.8	0.6	5	30
Fine ST2	80%	500	2	0.15	0.3	0.326	10	60
Coarse ST3	50%	500	2	0.4	0.8	0.6	5	30
Fine ST3	80%	500	2	0.3	0.5	0.15	10	60

noise. The resulting model would naturally not perform with high accuracy, but it adds sufficient vocabulary and serves as a baseline to improve upon in further phases. We call the resulting model as NMT_{Coarse}. NMT_{Coarse} including both the encoder and decoder is jointly trained to maximize the conditional log likelihood of the synthetic corpus as shown in Equation 3.

$$\max_{\theta_w} \frac{1}{N_w} \sum_{j=1}^{N_w} \log p_{\theta_w}(y_w^{(n)}|x_w^{(n)}) \qquad (3)$$

where $(y_w^{(n)}, x_w^{(n)})$ represents the $n-th$ sentence in the weak corpus of size N_w and θ_w denotes the set of all tunable parameters. The dropout and learning rate are kept high whereas the number of epochs is kept low since the primary motive for coarse learning is to learn only the general characteristics of the target language from the synthetic corpus, thereby making it easier to fine-tune the model. Detailed parameters used are provided in Table 4.
(Rapp and Vide, 2006) proposes a rule-based MT system using bigram dictionaries. As part of future work, this method can be employed in addition to our method to generate synthetic corpora for languages in which there is no existing MT tool available.

4.2 Fine-Tuning

This is the second and most important phase of our three-phase training approach. During this phase, a gold parallel corpus is needed.
This phase comprises of improving performance by fine-tuning the pre-trained model NMT_{Coarse} using the gold parallel corpus. This allows the model to be initialized with the weights learnt by the coarse model, rather than random weights.

In this phase, we employ the ILCI parallel corpus (with added linguistic features) for fine-tuning the pre-trained model - NMT_{Coarse}. This means that the low-data NMT model is not initialized with random weights, but with the weights learnt by the coarse model. The coarse model contains some amount of linguistic knowledge, in terms of lexical and semantic structure, word order and vocabulary. This information is imparted to the new model being trained using transfer learning. (Zoph et al., 2016) uses transfer learning to increase translation quality between resource scarce language pairs by incorporating the weights learnt during training for high resource language pairs. It was also found that languages having similar structure, like Fr $\longleftrightarrow$ En (French - English) showed better improvement in performance as compared to other languages having little similarity, like Uz $\longleftrightarrow$ En (Uzbek - English). Our approach is based on the intuition that transfer learning between the same language pair should perform better than its multilingual counterpart. Our experiments confirm this (Section 5) During fine tuning, the change in weights in each epoch learnt through transfer learning allows the model to align more towards the correct model. This is because the quality of the corpus employed during this phase is significantly better than the quality of the corpus employed for phase 1, i.e. Coarse Learning. However, since the size of this corpus is lesser, it is not a good idea to train the model directly on this corpus. This is

evident from the scores obtained by $Base_{NMT}$, the baseline NMT model trained only on the ILCI corpus.
We call the model generated after fine-tuning NMT_{FT}. Table 5 gives the results obtained by NMT_{FT}. Our experiments demonstrate that the quality of translation obtained using this technique is significantly better than SMT_{SA} as well as NMT_{Base}.
The hyper-parameters for training the model are carefully tweaked to achieve optimum performance. For example:
We use lower dropout and learning rate but considerably higher number of epochs in this phase as compared to Coarse Learning. This is done since the emphasis in this phase is to fine-tune the already learnt language characteristics and further learn new ones from the gold data.

Coarse Learning and Fine Tuning combined can be visualized as weakly supervised learning for our NMT model. Weak supervision is a technique of learning from noisy data or prior knowledge. (Haghighi and Klein, 2009) used rich syntactic and semantic features to induce prior knowledge for the task of coreference resolution. (Ratner et al., 2016) uses an ensemble of weak learners using rules to identify biomedical entities from medical documents.

4.3 Self-Training

Self-training is a form of semi-supervised learning, which is a technique of using both labelled and unlabelled data to improve the performance of a machine learning system. Self-training (Chapelle et al., 2009) involves iteratively classifying unlabelled data using a classifier trained on labelled data. The unlabelled data classified with highest confidence is used to further create the classifier along with the labelled data.

As part of the self-training stage, we generate a synthetic corpus using the fine-tuned model from the previous cycle. For example: NMT_{FT} from the first cycle is now used to translate the monolingual corpus rather than *Sampark* (Anthes, 2010). Coarse learning is then performed using this synthetic corpus as training data. This leads to better accuracy during coarse learning for the second cycle as compared to the previous iteration due to lesser noise in the synthetic corpus. The coarse model thus generated is again fine-tuned using the ILCI corpus. This forms one iteration of self-training. This entire cycle is repeated until there is minimal increase in translation accuracy.

This is an effective method specially when employed in the proposed three-phase training pipeline, since the quality of the synthetic data generator used during the first phase heavily influences the trasnlation accuracy. Since the fine-tuned model has a better quality than a rule-based or statistical MT system, we see significant gains on employing self-training.

The number of cycles to be performed for Self-Training (and in effect three-phase training) depends on the sizes of the monolingual corpus employed in the first phase as well as the parallel corpus employed in the second phase. If the latter is especially large in size, more self-training iterations can be performed. The size of our parallel corpus is 50,000 sentences. We perform three self-training iterations for our experiments since there was minimal to no increase in BLEU scores after that. The resultant model after three iterations is called NMT_{ST}. The results obtained by NMT_{ST} are given in Table 5 and discussed in Section 5.

Confidence estimation : OpenNMT (Klein et al., 2017) generates a prediction score for each translation, which is the cumulated log likelihood of the generated sequence. We use a threshold of -5.0 to filter out the low confidence translations. This ensures that the synthetic corpus employed for Coarse Learning in Training iteration 2 is of much better quality than the previous iteration. We observe improvement in scores by 2-5 percentage on employing this method, as opposed to using the same size of synthetic corpus in each training iteration.

5 Evaluation and Analysis

5.1 Results on the ILCI test set

We observe that although NMT models are good at learning language constructs from the parallel corpus itself, exploiting additional lin-

Table 5: Performance Comparison during various phases over ILCI test set in terms of BLEU scores

		urd	**pan**	**ben**	**guj**	**tam**
NMT_{FT}	**hin** ⇒	52.93	72.57	37.75	54.87	11.94
NMT_{ST}		53.95	73.71	38.77	55.52	12.27
NMT_{FT}	**hin** ⇐	60.22	73.2	38.97	54.64	22.04
NMT_{ST}		61.33	73.63	39.31	55.14	22.37

Table 6: Robustness comparison of models over different domains (in terms of BLEU scores)

		Housing				**Legal**			
		pan	**guj**	**urd**	**ben**	**pan**	**guj**	**urd**	**ben**
SMT_{SA}	**hin** ⇒	16.45	11.46	18.11	3.62	15.13	8.93	17.75	1.83
NMT_{Base}		17.48	13.23	19.53	4.74	16.42	11.35	19.02	2.89
NMT_{FT}		**23.71**	**17.62**	**24.49**	**13.22**	**22.27**	**14.07**	**25.41**	**7.7**
NMT_{ST}		22.69	16.92	22.23	11.03	19.13	13.29	23.69	6.1
SMT_{SA}	**hin** ⇐	13.85	12.73	14.78	3.0	12.45	11.93	15.63	2.72
NMT_{Base}		15.09	14.52	15.72	3.88	13.9	14.07	17.0	3.5
NMT_{FT}		**20.7**	**17.52**	**20.88**	**9.41**	**19.6**	**17.26**	**24.16**	**11.18**
NMT_{ST}		19.65	16.71	18.03	8.11	18.05	16.09	22.54	9.52

guistic information in the form of coarse learning - specially in low data conditions, provides further improvement in performance.

We can observe from Table 5 that a significant gain in scores is observed on employing three-phase training.

5.2 Results on test sets from different domains

We test the coverage of our model after three-phase training on test sets from different domains. We extract data samples from Housing and Legal domains respectively from the EMILLE parallel corpus (described in Section 3.1. We use these samples as test sets to evaluate the coverage of our models.

Table 6 shows the results obtained by SMT_{SA}, NMT_{Base}, NMT_{FT} and NMT_{ST} on test sets from different domains. We see improvement in accuracy as well as coverage - discussed below:

Two-phase vs. Three-phase Training : Accuracy vs. Coverage

Since the large monolingual corpus contains data from a variety of domains, NMT_{Coarse} develops a significantly big vocabulary, which leads to lesser number of Out of Vocabulary (OOV) words on out-of-domain data, as compared to NMT_{Base} and SMT_{SA}. The word order and lexical constructs learnt during coarse learning are retained and improved upon fine-tuning on the gold corpus.

NMT_{FT} exhibits best domain coverage results as can be seen from Table 6. This suggests that two-phase training obtains best results on out-of-domain data. Three-phase training includes self-training as well - it produces best results on in-domain data as can be observed from Table 5). Since the fine-tuned model is used to generate the synthetic corpus for the next self-training iteration, the quality of synthetic corpus thus obtained is higher than the one used during the previous iteration. Better synthetic data leads to better fine-tuning. This explains overall increase in accuracy after self-training over the ILCI test set. However, the coverage is affected a little. The reason can be attributed to a slight development of bias towards the health and tourism domains due to iterative fine-tuning. The domain coverage of three-phase training is still significantly better than SMT_{SA} and NMT_{Base}.

We conclude that the two-phase approach (Coarse Learning + Fine-Tuning) is more suitable for out-of-domain data, whereas the three-phase approach is better suited to translate in-domain data.

6 Conclusion

Data sparsity is a challenging problem in NMT, especially for resource-scarce language pairs. In this paper, we proposed an integrated approach to reduce the impact of data sparsity in NMT, using only little amount of parallel data. We demonstrated results using this approach on five Indian language pairs and showed a substantial improvement in translation quality. . We achieve comparative scores to the state-of-the-art for multiple language pairs. We propose that this is a effective method in the presence of an existing MT system and large monolingual corpora but inadequate parallel corpora. Future work includes using source as well as target translations for coarse learning and fine tuning, in addition to exploring methods for vocabulary compression. We would like to explore the application of this technique and its modifications for other resource-scarce languages, specifically the ones lacking a rule-based MT system. We would also like to evaluate the effectiveness of this approach in character-based translation. We would also like to experiment with using different atomic units for NMT, for eg. Orthographic syllables as units when dealing with translation among closely related languages OR subword-level units to ensure lesser number of Out-of-Vocabulary (OOV) words.

References

Gary Anthes. 2010. Automated translation of indian languages. *Communications of the ACM*, 53(1):24–26.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Junyoung Chung, Caglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *ICML*, pages 2067–2075.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1152–1161. Association for Computational Linguistics.

Bushra Jawaid, Amir Kamran, and Ondrej Bojar. 2014. A tagged corpus and a tagger for urdu. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*, number 39, page 413.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Anoop Kunchukuttan and Pushpak Bhattacharyya. 2016. Orthographic syllable as basic unit for smt between related languages. *arXiv preprint arXiv:1610.00634*.

Anoop Kunchukuttan, Abhijit Mishra, Rajen Chatterjee, Ritesh Shah, and Pushpak Bhattacharyya. 2014. Sata-anuvadak: Tackling multiway translation of indian languages. *pan*, 841(54,570):4–135.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of*

Computational Linguistics, pages 152–159. Association for Computational Linguistics.

Anthony McEnery, Paul Baker, Rob Gaizauskas, and Hamish Cunningham. 2000. Emille: Building a corpus of south asian languages. *VIVEK-BOMBAY-*, 13(3):22–28.

Reinhard Rapp and Carlos Martin Vide. 2006. Example-based machine translation using a dictionary of word pairs. In *Proceedings, LREC*, pages 1268–1273.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, pages 3567–3575.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Interspeech*, pages 194–197.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Paul J Werbos. 1990. Backpropagation through time, what it does and how to do it. *Proceedings of the IEEE*, 78.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1604.02201*.

Reference Scope Identification for Citances Using Convolutional Neural Network

Saurav Jha
MNNIT Allahabad, India
mail@sauravjha.com.np

Aanchal Chaurasia
NIT Rourkela, India
aanchal21194@gmail.com

Akhilesh Sudhakar
IIT (BHU), Varanasi, India
akhileshs.s4@gmail.com

Anil Kumar Singh
IIT (BHU), Varanasi, India
nlprnd@gmail.com

Abstract

In the task of summarization of a scientific paper, a lot of information stands to be gained about a reference paper, from the papers that cite it. Automatically generating the reference scope (the span of cited text) in a reference paper, corresponding to citances (sentences in the citing papers that cite it) has great significance in preparing a structured summary of the reference paper. We treat this task as a binary classification problem, by extracting feature vectors from pairs of citances and reference sentences. These features are lexical, corpus-based, surface and knowledge-based. We extend the current feature set employed for reference-citance pair identification in the current state-of-the-art system. Using these features, we present a novel classification approach for this task, that employs a deep Convolutional Neural Network along with two boosting ensemble algorithms. We outperform the existing state-of-the-art for distinguishing between cited spans and non-cited spans of text in the reference paper.

1 Introduction

Citation sentences or 'citances' that cite a reference paper (RP) can give valuable information about the larger context in which the RP is written, key ideas behind the RP and a concise synopsis of it. All of this is important for a task like scientific paper summarization, which not only requires the content of a paper but also meta-information about it. This kind of information would otherwise have to be obtained from sources such as literature reviews and surveys about the paper, which in turn is time-consuming and labor-intensive. This goal has also been outlined in a recent shared task on scientific paper summarization, the 3rd Computational Linguistics Scientific Document Summarization Shared Task[1].

The first step towards building a system that can obtain information about an RP from a citing paper (CP) that cites it, is to find spans of text in the RP that are cited by a particular citance in the CP. In the context of the above-mentioned shared task, this first step is referred to as Task 1A. Task 1A, thus offers a good foundation for the goal mentioned above, by identifying the relevant reference sentences for a citance. We present a novel approach to Task 1A. While we build on previous work by Yeh et al. (2017), our major contributions can be described as:

- We model a new feature set to represent a citance-reference sentence pair along with building a classification system that uses a binary classification technique for classifying a <CP sentence, RP sentence> pair according to whether the CP sentence cites the RP sentence or not.
- We show performance gains over the results of Yeh et al. (2017)(which is the current state-of-the-art) by achieving better F1-scores, using a feature set that has lesser number of features than that used in the above work.
- We explore various measures for evaluating similarity between texts while build-

[1] http://wing.comp.nus.edu.sg/ cl-scisumm2017/

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 23–32

ing this feature set. Feature representations extracted (as described later), are used to train three binary classifiers - an Adaptive Boosting Classifier (ABC), a Gradient Boosting Classifier (GBC) and a CNN classifier.

The datasets provided for this year's as well as last year's shared task have been used.

2 Related Work

There has been a large amount of work on the task of summarizing scientific documents. However, as is clear from review surveys and papers such as Jones (2007), Teufel and Moens (2002) and Nenkova (2011), just using citances of a paper does not taken into account the context of a user or place the paper in a larger perspective of related work. Most of the related work on the task of identifying text spans in the RP that correspond to a particular citance, have been presented at the shared task mentioned in the previous section. We highlight some relevant work and various methods used for this task.

Yeh et al. (2017) also used a binary classification approach for Task 1A, as we do. They used five classification algorithms to learn the binary classification model, with L2-SVM performing the best. Moraes et al. (2016) used SVM with subset tree kernel, a type of convolution kernel. They computed similarities between three tree representations of the citance and reference text. Li et al. (2016) used an SVM classifier with a topical lexicon to identify the best matching reference spans for a citance, using IFD similarity, Jaccard similarity and context similarity. The PolyU system by Cao et al. (2016), for Task 1a, used SVM-rank with lexical and document structural features to rank reference text sentences for every citance. Klampfl et al. (2016) applied a modified version of an unsupervised summarization technique (TextSentenceRank) to the reference document. Nomoto (2016) treated the problem as a ranking problem, learning one component of the similarity through a neural network and using TF-IDF scores for the other component. Aggarwal and Sharma (2016a) employed lexical and syntactic dependency cues in writing rules to extract text spans in the RP for a given CP citance. Malenfant and Lapalme (2016) presented a novel approach to solve this task. They first performed another task of identifying the facet of each sentence of the RP. These facets belonged to a predefined set of facets, such as *introduction, abstract, results, etc.* They then used the facet information to match each sentence to a citance having the same facet in the CP.

3 Method

The structure of the dataset is described in Section 4.1. Citances and their actual reference texts are extracted from gold-standard annotations. Citances in CPs are paired with each sentence in the RPs, along with a binary label indicating their actual reference relations - 0 if the citance actually refers to the RP sentence and 1 if it doesn't. For each such pair, a feature vector is extracted that describes the relatedness between the given citance and the reference sentence. These feature vectors, along with their corresponding labels, are used to train the binary classifiers.

3.1 Feature Extraction

As mentioned in the section on related work, most approaches to this task have either been based on ranking of possible cited sentences in the RP for a given CP citance, or on binary classifying each RP sentence as relevant or irrelevant to a given CP citance. We use the latter approach. Our method is based on the assumption that a CP sentence and corresponding RP sentence should be semantically and lexically similar, representing similar meaning, idea or abstract concept. This is a natural assumption to make, since modeling the problem based on this assumption helps to separate relevant sentences (to the CP citance) in the RP from irrelevant ones. Inspired by the idea of Yeh et al. (2017), the feature set for each citance-reference pair is divided into three different *classes* of citation-dependent features (i.e., lexical, knowledge-based and corpus-based) and one *class* of citation-independent features (i.e., surface). However, we must mention here that our work is significantly dif-

ferent from Yeh et al. (2017), when it comes to the set of features used. Through control experiments (Section 5.1), we show the effect of using our set of feature over theirs. We incorporate several modified and newly added features.

The features marked by an asterisk (*) are the ones that are borrowed, but modified. The features marked by two asterisks (**) are the newly added features in this work. For features that have been borrowed from Yeh et al. (2017), more elaborate details about them can be seen in their work.

3.1.1 Lexical features

This class deals with the features representing similarity measure of words for each pair of citance and reference sentence. As suggested by the results of Kenter et al. (2016) for short text similarity tasks, the overall sentence similarity measure based on each feature is calculated by averaging the similarities over all the words in the sentences.

1. **Word overlap*:** Word overlap between the citance and the reference sentence based on five metrics: *Dice coefficient, Jaccard coefficient, Cosine similarity, Levenshtein distance based fuzzy string similarity and modified gestalt pattern-matching based sequence matcher score*, the last one as reported by Ratcliff and Metzener (1988).

2. **TF-IDF similarity:** The TF-IDF vector cosine similarity between the citance and the reference sentence as reported by Baeza-Yates and Ribeiro-Neto (2011).

3. **ROUGE measure:** The ROGUE (Lin and Hovy, 2003) metrics used are: ROGUE-1, ROGUE-2 and ROGUE-L.

4. **Named entity overlap*:** Measured using Dice coefficient, fuzzy string similarity, sequence matcher score and word2vec similarity.

5. **Number overlap*:** Number overlap between the citance and the reference sentence measured by fuzzy string similarity and sequence matcher score.

6. **Significance of citation-related word pairs:** The number of significant word pairs and the summation of significance scores of such word pairs extracted for each citance-reference pair based on Pointwise Mutual Information (PMI) score (Church and Hanks, 1989) collected from a dictionary containing significant words pairs appearing in the cited citance-reference pairs.

3.1.2 Knowledge-based features

1. **WordNet-based semantic similarity*:** The best semantic similarity score between words in the citance and the reference sentence out of all the sets of cognitive synonyms *(synsets)* present in the WordNet, following Miller (1992) and Pedersen et al. (2004).

3.1.3 Corpus-based feature

1. **Word2Vec-based semantic similarity**:** The word-to-word semantic similarity between the citance and the reference sentence is obtained based on the pre-trained embedding vectors of the GoogleNews corpus, following Mikolov et al. (2013). Campr and Jezek (2015) show several advantages that such embeddings offer, compared to those generated by traditional algorithms, such as LSA.

3.1.4 Surface features

This class includes features dealing primarily with the morphology of the reference sentences. These include:

1. **Count of words:** The count of words in the reference sentence.

2. **Count of characters**:** The total count of all characters in the reference sentence.

3. **Count of numbers:** The count of numbers in the reference sentence.

4. **Count of special characters**:** The number of special characters in the reference sentence : *“@”, “#”, “$”, “%”, “&”, “*”, “-”, “=”, “+”, “>”, “<”, “[”, “]”, “{”, “}”, “/”.*

5. **Normalized count of punctuation markers******:** The ratio of count of punctuation characters to the total count of characters in the reference sentence.

6. **Count of long words******:** The count of words in the reference sentence exceeding six letters in length.

7. **Average word Length******:** The ratio of count of total characters in a word to the count of words in the reference sentence.

8. **Count of named entities:** The number of named entities in the reference sentence.

9. **Average sentiment score******:** The overall positive and negative sentiment score of the reference sentence averaged over all the words, based on the SentiWordNet 3.0 lexical resource as described by Baccianella et al. (2010).

10. **Lexical richness******:** The lexical richness of the reference sentence based on Yule's K index.

3.2 Classification Algorithms

As our approach treats the training data as pairs of citances and reference sentences, the number of reference sentences that a citance refers to is much smaller for a reference paper, leading to a highly imbalanced data set with the ratio of non-cited to cited pairs being 383.83 : 1 in the combined corpus of development and training set and 355.76 : 1 in the test set corpus. This is not surprising since CPs usually cite only a small text span of an entire RP. Hence, our dataset is hugely imbalanced with negative examples being the majority. Following the work of Bowyer et al. (2002), we experimented with combinations of three different degrees of Random under-sampling (20%, 30% and 35%) on the majority class (negative samples). On each undersampled dataset, we apply the SMOTE (Synthetic Minority Over-sampling Technique) method (Bowyer et al. (2002)) to generate synthetic cited pairs until the ratio of the cited to non-cited pairs is 1:1. The best results were obtained with 30% random undersampling rate. To take care of correlated features, if any, Principal Component Analysis (PCA), following Tipping and Bishop (1997) is applied on both training and testing feature sets. Experiments were done by varying the number of principal components from 30-40 and the best performance was obtained by retaining the top 35 principal components.

For the classification task, we use two boosting ensemble algorithms: Adaptive Boosting Classifier (ABC) as described by Abe et al. (1999), Gradient Boosting Classifier (GBC) as described by Friedman (1999) and a deep Convolutional Neural Network (CNN) as described by Schmidhuber (2015).

The implementation of ABC and GBC rely on sci-kit learn[2], while the CNN is implemented using Keras[3] (Chollet et al. (2015)). Gensim (Rehurek and Sojka (2010)) is used to carry out word2vec related operations.

3.2.1 Boosting Ensemble Algorithms

Boosting ensemble algorithms work by creating a sequence of models that attempt to correct the mistakes of the models used before them in the sequence. Therefore, these offer the added benefit of combining outputs from weak learners (those whose performance is at least better than random chance) to create a strong learner with improved prediction performance, by paying higher focus on instances that have been misclassified or have higher errors by preceding weak rules. This is assisted by a majority vote of the weak learner's predictions weighted by their individual accuracy. Figure 1 shows the illustration of such a boosting framework, as described by Bishop and Nasrabadi (2007).

The **Adaptive Boosting Classifier (ABC)** algorithm works in a similar way discussed above. The base classifier (or weak learner) used in this case is a decision tree.

Gradient Boosting Classifier (GBC), on the other hand, begins by training several models sequentially on the original data set while allowing each model to gradually minimize the loss function of the whole system using the Gradient Descent method, as described by Collobert et al. (2004). The

[2]http://scikit-learn.org
[3]https://keras.io

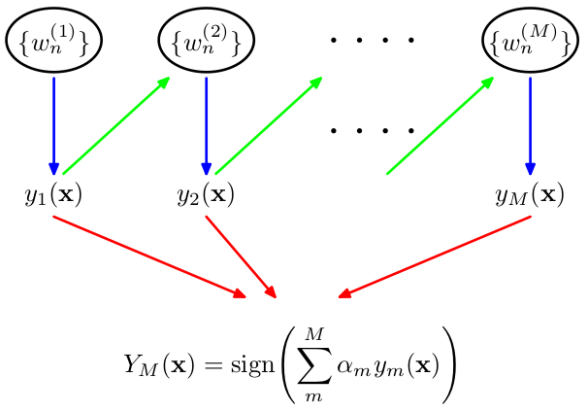


Figure 1: Schematic illustration of the boosting framework. Adapted from Bishop and Nasrabadi (2007): each base classifier $y_m(x)$ is trained on a weighted form of the training set (blue arrows) in which the weights $w_n(m)$ depend on the performance of the previous base classifier $y_m{-}1(x)$ (green arrows). Once all base classifiers have been trained, they are combined to give the final classifier $Y_M(x)$ (red arrows)

base classifiers in a GBC are regression trees. Since our task is a binary classification, only one regression tree is used as a special case.

3.2.2 Convolutional Neural Network

Convolutional Neural Networks, as described in Schmidhuber (2015), have the ability to extract features of high-level abstraction with minimum pre-processing of data. They have been widely used for sentence classification problems, such as by Kim (2014). Recently, Ngoc Giang et al. (2016) also used CNNs for a sequence classification problem involving classification of DNA sequences by considering these sequences as text data. Given the success of CNNs on these, we explore their use in our task.

However, in our case, a class-imbalance problem occurs due to the number of positive reference-citance instances being far too low *(495)*. These are too few examples for any deep learning model to extract meaningful features from the original text. Using the original sentences, and modeling it directly as a sequence classification on pairs of sentences would introduce too much sparsity owing to this imbalance. Not surprisingly, our experiments on using original sentences directly in an attention-based RNN model (2015) resulted in a precision score of 0.002 for positive and 0.24 for negative samples. Thus, we choose to train the CNN on the feature sets as inputs instead of the sentences directly. Figure 2 describes the CNN architecture chosen by us after repeated experiments and tuning on the development data.

A 1D Convolutional layer accepts inputs of the form *(Height * Width * Channels)*. In our case, we can visualize each feature vector as an image with a unit channel, unit height and a width equal to the number of features in the reduced feature vector obtained after applying PCA. Therefore, the input shape for the vector to be fed into the input layer of the CNN, becomes *(No. of features * 1)*.

3.3 Post Filtering

The binary classifier may classify multiple sentences in the RP as positive, i.e., being relevant to a particular citance. However, the existence of inherent errors in the model means that all of these sentences may not be in the ideal text span of the RP corresponding to the citance. In order to reduce our false positive error rate, we post-process by filtering out some of these false positives. We use the approach of Yeh et al. (2017) for the post-filtering task. In this approach, the final output denotes the top-k sentences from the ordered sequence of classified reference sentences based on the TF-IDF vector cosine similarity score to measure the relevance between the citance and the reference sentences. All sentences other than the top-k are not included in the final output text span, even though the model might have labelled them as positive.

4 Experiments

4.1 Dataset

We use the development corpus, the training corpora and the test corpus provided for the CL-SciSumm Shared Tasks 2016[4] and 2017[5]. As reported in Jaidka et al. (2016), each corpus comprises 10 reference articles, their citing papers and annotation files for each reference article. The citation annotations specify

[4]http://wing.comp.nus.edu.sg/cl-scisumm2016/
[5]http://wing.comp.nus.edu.sg/ cl-scisumm2017/

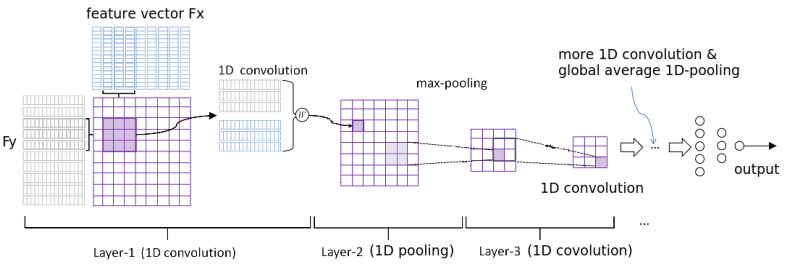


Figure 2: Our CNN architecture: stack of two 1-D convolutional layers with 64 hidden units each (ReLu activations) + 1-D MaxPooling + stack of two 1-D convolutional layers with 128 hidden units each (ReLu activations) + 1-D Global Average Pooling + 50% Dropout + a single unit output dense layer (sigmoid activation)

citances, their associated reference text and the discourse facet that it represents.

4.2 Experimental Settings

Precision, Recall and F1-Score are used as evaluation metrics. The average score on all topics in the test corpus is reported. We run experiments on two separate training sets.

In the **first run**, we use data only from the 2016 shared task, and not from the 2017 shared task. This is because we need a common ground for comparison with the existing state-of-the-art (Yeh et al. (2017)), which used this dataset. We first train our data on the training set, and tune the CNN's hyperparameters on the development set. We then augment the training data and the development data to train the final models. We test our model on the test provided as part of this dataset. Table 2 shows the performance of the CNN model on this test set, and compares it with the existing state-of-art and another well-performing model. We have reported only the CNN's performance in this table as (as will be seen in the results of the second run), this is a better performing model than ABC and GBC, in our experimental setup.

In the **second run**, we make use of the datasets from both 2016 and 2017. Both the training datasets are augmented to form the initial training set. After tuning the CNN's hyperparameters on the development set (which is the same for both 2016 and 2017), the initial training and development sets are augmented to form the final training set. Grid search algorithm, as given by Bergstra and Bengio (2012), over 10-fold cross validation is used to find the best model parameters for ABC and GBC listed in Table 1. Since the gold-standard annotations for the 2017 test set were not yet available at the time of conducting our experiments, we use only the test set of 2016. We report performance of ABC, GBC as well as the CNN classifier on this test set. Table 3 shows these results.

Table 1: **Model parameter settings**

Classifier	**Architecture and Parameter settings**
ABC	Learning rate for shrinking the contribution of each decision tree = **1.3**, Boosting algorithm = **SAMME.R** for faster convergence
GBC	Learning rate for shrinking the contribution of regression tree = **0.15**, Loss = **Deviance** for probabilistic outputs, No. of Boosting stages = **100**

5 Results and Analysis

Precision, recall and F1-score obtained by the models on the test set with respect to the positive classes, evaluated by 10-fold cross validation are shown in Table 3. The CNN-based classifier was trained for 30 epochs. The best scores for each metric have been shown in bold.

Table 2 shows a comparison of the F1-score achieved by our model with that of the pre-

Table 2: **F1 scores of previous models**

System	**F1-scores**
Yeh et al. (2017)	0.1443
Aggarwal and Sharma (2016b)	0.11
Our Method	**0.2462**

vious models used for the task. The L2-SVM system by Yeh et al. (2017) produced an F1-score of 0.1443, which is the highest reported yet to the best of our knowledge. Our model outperforms it in terms of F1-score. It must be mentioned here that Klampfl et al. (2016) reported an F1-score of 0.346 on the development set corpus and 0.432 on the training set corpus of 2016. However, we have not considered their system in Table 2 because of the unavailability of their performance results on the test set corpus. Figure 3 compares the performance of our CNN classifier with their TextSentenceRank assisted sentence classifier on the development and training set corpus *(80:20 train:test split)* of 2016. Although the CNN classifier performs better on both the corpora, the improvement on the development corpus is much more significant than that on the training.

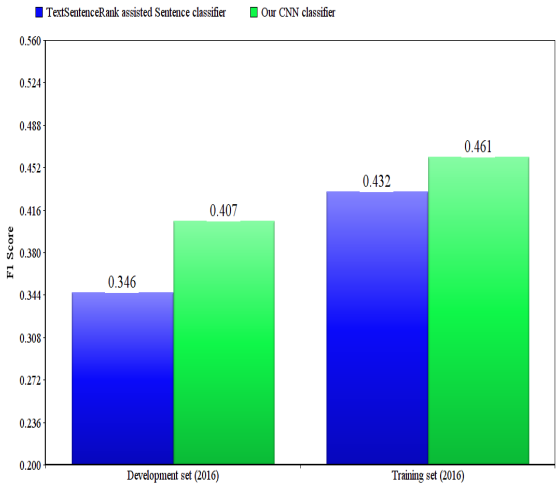


Figure 3: A comparison of the performance of our CNN-based classifier on the development and training set of 2016 with that of Klampfl et al. (2016)

5.1 Control Studies

We run control studies to analyze the contribution of each class of features to our final performance. We also control for the different techniques used, such as SMOTE and PCA, to see their effect. These control studies also help us to understand why our model outperformed the previous state-of-art. Since the CNN is our best performing classifier, we make use of it to perform these control studies.

5.1.1 Effect of feature classes

Figure 4 shows the effect of each class of features. We obtain these graphs by removing one class of features each time from the feature set and calculating the performance using all the other classes. From the bar plot in Figure 4, it is apparent that the class of lexical features contributes the most in distinguishing between a positive and a negative example. Not using this class of features gives 0.3371 lesser F1-score than when using all the features. This means that an information retrieval based component to this problem using lexical features such as TF-IDF, ROUGE etc. as mentioned in section 3.1.1 is the most important for this task. It is also possible that this class shows the maximum effect because of the good number of features in this class, i.e., 6. The second most significant class of features is the class of corpus-based features. Not using this class of features gives 0.3002 lesser F1-score than when using all the features. Our class of corpus-based features has just the word2vec feature. It is not surprising that this feature shows a high impact because word2vec representations capture a good level of semantic and syntactic similarity, which was one of the assumptions we built the model on. Not using the class of surface features gives 0.2429 lesser F1-score than when using all the features. One reason for the impact of surface features could be that it is perhaps the only class of features that takes numbers and special characters into account, and these are significantly high in number in scientific pa-

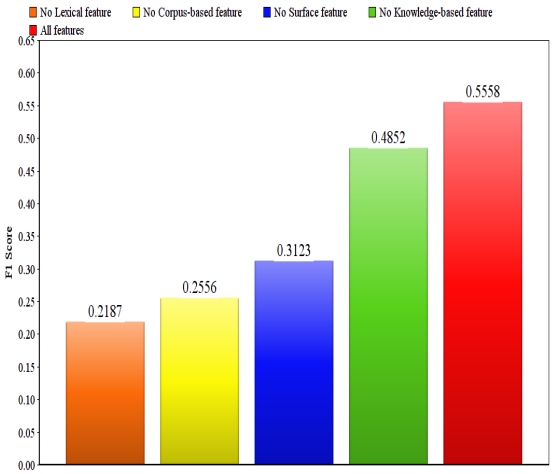


Figure 4: F1 Scores of CNN Model with different feature selection settings

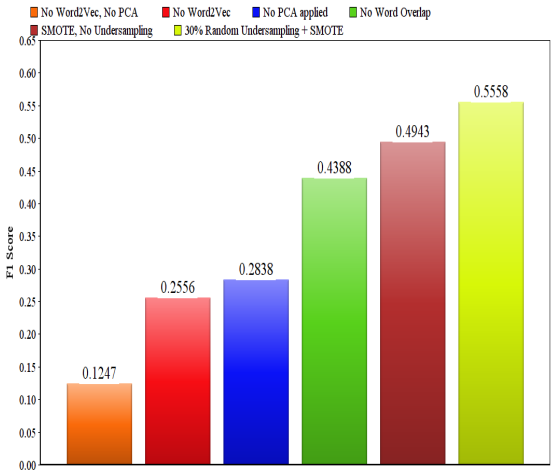


Figure 5: F1 Scores of CNN Model with different feature selection settings

pers. This class also has a high number of features, i.e., 10. Not using the class of surface features gives 0.0706 lesser F1-score than when using all the features. The plot also shows that the impact of WordNet-based features contribute the least to distinguishing between positive and negative examples.

It might therefore be concluded that part of the reason why our model outperforms the state-of-the-art is that their model does not make use of word2vec, while we do so. It also appears that the modified lexical features that we have used, namely, named entity overlap, number overlap and word overlap provide an added advantage to our model, over the state-of-the-art.

5.1.2 Effect of data-handling techniques

Figure 5 shows the contribution of different pre-processing and processing techniques used such as SMOTE (oversampling), undersampling and dimensionality reduction using PCA. There are a few observations that can be drawn from the bar plot in Figure 5. Firstly, dimensionality reduction is a crucial important step in this task. When PCA was not used on the feature set, the performance dropped from 0.5558 to 0.2838, which is a reduction in F1-score of 0.2720. The existing state-of-the-art has a higher number of features than our work does, and does not perform dimensionality reduction on these features, which might be one of the reasons behind the better performance achieved in our work. Further, we see that oversampling using SMOTE gives an improvement of 0.0555 and using undersampling over and above this further improves the performance by 0.0615.

Table 3: **Results obtained by different models**

Model	**Precision**	**Recall**	**F1-score**
ABC	0.7141	0.3579	0.4925
GBC	**0.7439**	0.3237	0.4512
CNN	0.6556	**0.5973**	**0.5558**

5.1.3 Classifier-wise performance

Table 3 shows the performance on the combined dataset of the 2016 and 2017 versions of the shared task, as described in more detail in section 4.2 as the 'second run'. The CNN model gives the best performance on recall and F1-score, while the GBC model gives the best precision. Precision for each classifier is considerably higher than that of recall, indicating that there are relatively few

false positives while a significant number of true positives have been missed out.

6 Future Work

To our knowledge, this is the first attempt which has used a deep learning model for addressing the task. However, for training our model, we had to be entirely dependent on the feature sets extracted. The number of positive instances in the corpus provided is still too low to train the model using the conventional CNN sequence-to-sequence approach, which, given more data, might be able to learn more interesting patterns in the citance-reference pairs. Also, recent extensions to word2vec such as the **Paragraph Vector** (Le and Mikolo (2014)) can be used to further enhance the semantic similarity measures between the reference-citance pairs.
Furthermore, the binary labels assigned to each <CP sentence, RP sentence> pair can be used to establish some partial order in between the training instances, which in turn, can help in modeling the task as a Learning to rank problem. This ordering can then be incorporated to predict the relevance-based ranking of referenced sentences for a citance.

7 Conclusions

We describe our work on reference scope identification for citances using an extended feature set applied to three different classifiers. Among the classifiers trained to distinguish cited and non-cited pairs, the CNN-based model gave the overall best results with an F1 score of 0.5558 on the combined corpus of CL-SciSumm 2016 and 2017. We also achieved an F1 score of 0.2462 on the 2016 dataset, which surpasses the previous state-of-the-art accuracy on the dataset. In addition to this, we carry out control studies reporting the contribution of various feature classes as well as feature selection methods that have been used by us.

References

Naoki Abe, Yoav Freund, and Robert E. Schapire. 1999. A short introduction to boosting.

Peeyush Aggarwal and Richa Sharma. 2016a. Lexical and syntactic cues to identify reference scope of citance. In *BIRNDL@ JCDL*, pages 103–112.

Peeyush Aggarwal and Richa Sharma. 2016b. Lexical and syntactic cues to identify reference scope of citance. In *BIRNDL@JCDL*.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*.

Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. 2011. Modern information retrieval - the concepts and technology behind search, second edition.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Christopher M. Bishop and Nasser M. Nasrabadi. 2007. Pattern recognition and machine learning. *J. Electronic Imaging*, 16:049901.

Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. Smote: Synthetic minority oversampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357.

Michal Campr and Karel Jezek. 2015. Comparing semantic models for evaluating automatic document summarization. In *TSD*.

Ziqiang Cao, Wenjie Li, and Dapeng Wu. 2016. Polyu at cl-scisumm 2016. In *BIRNDL@JCDL*.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Ronan Collobert, Patrick Gallinari, Léon Bottou, Hélène Paugam-Moisy, Samy Bengio, and Yves Grandvalet. 2004. Large scale machine learning.

Jerome H. Friedman. 1999. Greedy function approximation: A gradient boosting machine.

Kokil Jaidka, Muthu Kumar Chandrasekaran, Sajal Rustagi, and Min-Yen Kan. 2016. Overview of the cl-scisumm 2016 shared task. In *BIRNDL@JCDL*.

Karen Spärck Jones. 2007. Automatic summarising: The state of the art. *Information Processing & Management*, 43(6):1449–1481.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. *CoRR*, abs/1606.04640.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Stefan Klampfl, Andi Rexha, and Roman Kern. 2016. Identifying referenced text in scientific publications by summarisation and classification techniques. In *BIRNDL@JCDL*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Lei Li, Liyuan Mao, Yazhao Zhang, Junqi Chi, Taiwen Huang, Xiaoyue Cong, and Heng Peng. 2016. Cist system for cl-scisumm 2016 shared task. In *BIRNDL@JCDL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Bruno Malenfant and Guy Lapalme. 2016. Rali system description for cl-scisumm 2016 shared task. In *BIRNDL@ JCDL*, pages 146–155.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

George A. Miller. 1992. Wordnet: A lexical database for english. *Commun. ACM*, 38:39–41.

Luis Moraes, Shahryar Baki, Rakesh M. Verma, and Daniel Lee. 2016. University of houston at cl-scisumm 2016: Svms with tree kernels and sentence similarity. In *BIRNDL@JCDL*.

Ani Nenkova, Kathleen McKeown, et al. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233.

Nguyen Ngoc Giang, Vu Anh Tran, Duc Luu Ngo, Dau Phan, Favorisen Lumbanraja, M Reza Faisal, Bahriddin Abapihi, Mamoru Kubo, and Kenji Satou. 2016. Dna sequence classification by convolutional neural network. 09:280–286, 01.

Tadashi Nomoto. 2016. Neal: A neurally enhanced approach to linking citation and reference. In *BIRNDL@ JCDL*, pages 168–174.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet: : Similarity - measuring the relatedness of concepts. In *AAAI*.

John W. Ratcliff and David Metzener. 1988. *Pattern Matching: The Gestalt Approach*.

Radim Rehurek and Petr Sojka. 2010. Software framework for topic modelling with large corpora.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks : the official journal of the International Neural Network Society*, 61:85–117.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Computational linguistics*, 28(4):409–445.

Michael E. Tipping and Christopher M. Bishop. 1997. Probabilistic principal component analysis.

Jen-Yuan Yeh, Tien-Yu Hsu, Cheng-Jung Tsai, and Pei-Cheng Cheng. 2017. Reference scope identification for citances by classification with text similarity measures. In *ICSCA '17*.

A vis-à-vis evaluation of MT paradigms for linguistically distant languages

Ruchit Agrawal
LTRC
IIIT Hyderabad

Jahfar Ali
LTRC
IIIT Hyderabad

Dipti Misra Sharma
LTRC
IIIT Hyderabad

Abstract

Neural Machine Translation is emerging as the de facto standard for Machine Translation across the globe. Statistical Machine Translation has been the state-of-the-art for translation among Indian languages. This paper probes into the effectiveness of NMT for Indian languages and compares the strengths and weaknesses of NMT with SMT through a vis-a-vis qualitative estimation on different linguistic parameters. We compare the outputs of both models for the languages English, Malayalam and Hindi; and test them on various linguistic parameters. We conclude that NMT works better in most of the settings, however there is still immense scope for the betterment of accuracy for translation of Indian Languages. We describe the challenges faces especially when dealing with languages from different language families.

1 Introduction and Related Work

There is an immense scope in the development of translation systems which cater to the specific characteristics of languages under consideration. Indian languages are not an exception to this, however, they add certain specifications which need to be considered carefully for effective translation. Firstly, they span across multiple language families like the Indo-Aryan and Dravidian languages. Secondly, there is a lack of large parallel corpora for most of these languages, which are required to build robust systems by the SMT and NMT paradigms.

This paper probes in to the competence of different MT paradigms with respect to language pairs which belong to different language families. Dravidian languages raise many intriguing issues in modern linguistics. One of them is the differentiation of the finiteness and non finiteness of clauses with its tense inflection in verbs (Amritavalli, 2014), (McFadden and Sundaresan, 2014), (Tonhauser, 2015). Scrambling effect on canonical word order (Jayaseelan, 2001) is another such phenomenon. It is to be observed when dealing with complex syntactical structures containing cleft constructions in Malayalam (Jayaseelan and Amritavalli, 2005).

Relative clause structures, nominal clause structures and their coordination constructions in Dravidian languages are other interesting phenomena (Amritavalli, 2017), (Jayaseelan, 2014). The analysis made in the paper describes the handling of all these linguistic phenomena in the context of machine translation.

Neural Machine Translation is emerging as a de facto standard for Machine Translation across the globe. However, a manual inspection of the output translations reveals significant scope for improvement in translation quality. We perform a comparative analysis of Neural and Phrase-based Statistical MT techniques and highlight the strengths and weaknesses of each paradigm with respect to handling of different linguistic phenomena. The enquiry throws light on some of the challenging cases encountered when translating between morphologically rich and free word order languages and the other end of morphologically less complicated and word order specific languages. A set of basic observations are made after extensive testing of SMT and NMT outputs on these language pairs. We observe that NMT performs better than SMT for most of the linguistic phenomena considered. However; one of the major hurdles to

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 33–42

deliver the correct output between morphologically rich languages to to morphologically weak languages is the inadequacy of NMT to generate word forms with correct affixes.

The analysis can generate fruitful insights in the modification of NMT / SMT based techniques to generate efficient results. The insights can be taken into consideration during the building of parallel corpora in the future or using linguistic features as additional informaiton for training NMT models. The analysis also enables the usage of a particular paradigm depending upon the language pair and domain in consideration.

2 Motivation

2.1 Characteristics of Indian languages

The majority of Indian languages are morphologically rich and depict unique characteristics, which are significantly different from languages such as English. Some of these characteristics are the relatively free word-order with a tendency towards the Subject-Object-Verb (SOV) construction, a high degree of inflection, usage of reduplication, conjunct verbs, relative participal forms and correlative clause constructions. These unique characteristics coupled with the caveats of evaluation metrics described in Section 2.3 pose interesting challenges to the field of Indian Language MT - both in terms of development of efficient systems as well as their evaluation.

For example, in Hindi, a sentence s containing the words w_1,w_2,..,w_n can be formulated with multiple variants of word ordering. This behavior is depicted in Table 1, which shows two Hindi translations of the following English sentence :
'Shyam has given the book to Manish.' Although they use different word-order, both of them are semantically equivalent and correct translations of the source sentence.
Similarly, for the sentence 'The sun has set', there can be multiple valid translations, as shown in Table 2. It can be noted that 'सूर्य ' and 'सूरज ' are synonyms of 'Sun' in Hindi.

In addition to these, there are many subtle differences in the ways different Indian languages encode information. For example, Hindi has two genders for nouns whereas Gujarati has three. There are also many ambiguities introduced in language (both at lexical as well as sentence levels) due to the socio-cultural reasons and partial encoding of information in discourse scenario. In addition to this, the majority of Indian languages encode a significant amount of linguistic information in their rich morphological structures, and often lexemes can have multiple senses. All these factors like linguistic conventions, socio-cultural knowledge, context and highly inflectional morphology combined together make Indian languages a challenging terrain for Machine Translation.

2.2 Variation in linguistic constructions in IA and DR languages

Even though Indian languages are all typologically SOV, there are distinct syntactic peculiarities in Dravidian languages (DR) that makes MT challenging between Indo-Aryan (IA) and Dravidian languages. Two such phenomena are shown by the examples below:

1.
   - Hindi Sentence : राम ने बोला कि वह घर जा रहा था
   - Transliteration : rām nē bōl-ā ki vah ghar jā rahā thā
   - Gloss : Ram ERG tell-PST S.CONJ 3.SG.D.PRON home go AUX1-CONT AUX2-PST
   - Meaning : Ram said that he is going home

2.
   - Malayalam Sentence : അവൻ വീട്ടിലോക്ക് പോകുകയാണ് എന്ന് രാമൻ പറഞ്ഞു
   - Transliteration : avan vīṭṭilēākk pēākukayāṇ enn rāman paṟaññu
   - Gloss : He-NOM home-LOC-towards go-INF COP QT Raman-NOM say-PST.

3.
   - Telugu Sentence : రాముడు తాను ఇంటికి వెళ్తున్నట్టుగా చెప్పాడు

Table 1: Different Hindi translations corresponding to the English sentence - “Shyam has given the book to Manish.” (Due to word order)

	Hindi	Transliteration
Sent : 1	मनीष को श्याम ने किताब दे दी ।	maneesh ko shyaam ne kitaab de dee
Sent : 2	श्याम ने मनीष को किताब दे दी ।	shyaam ne maneesh ko kitaab de dee

Table 2: Two different translations corresponding to the English sentence - “The sun has set.” (Due to many-to-many mapping between vocabulary)

	Hindi	Transliteration
Sent : 1	सूर्य डूब चुका है ।	soorya doob chuka hai
Sent : 2	सूरज डूब चुका है ।	sooraj doob chuka hai

- Transliteration : rāmuḍu tānu iṇṭi-ki veḷ-tunn-aṭṭugā cepp-ā-ḍu
- Gloss : Ram 3P.REFL.PRON home-DAT go-PRES-MANNER.ADV tell-PST-3.M.SG

4.
- Tamil Sentence : ராமன் தான் வீட்டுக்கு செல்வதாக கூறினான்
- Transliteration : rāman̲ tān̲ vīṭṭu-kku cel-vat-āka kūṟ-in̲-ān̲
- Gloss : Ram 3P.REFL.PRON home-DAT go-NPST.R.PART-MANNER.ADV tell-PST-3.M.SG

Above example shows that in Hindi the main clause is followed by subordinate clause and both the clauses are connected by a subordinating conjunction ‘ki’. For Malayalam, The embedded clausal structure with quotative particle ‘ennu’ is the only kind of sentence possible to have two finite verbs (Asher and Kumari, 1997). In Telugu and Tamil (Dr), the subordinate clause is embedded within the main clause and connection between them is established morphologically through adverbial inflections or sometimes a quotative marker is used to connect the two clauses. These phenomena explain the relatively lower performance on Dravidian languages as compared to Indo-Aryan languages.

2.3 Challenges in automatic evaluation

A key aspect in developing efficient MT systems is addressing the issue of effective metrics for automatic evalution of translations, since manual evaluation is expensive and time-consuming. There has been significant interest in this area, both in terms of development as well as evaluation of MT metrics. The Workshop on Statistical Machine Transla- tion (Callison-Burch et al., 2007; Callison-Burch et al., 2008; Callison- Burch et al., 2009) and the NIST Metrics for Machine Translation 2008 Evaluation 1 have both collected human judgement data to evaluate a wide spectrum of metrics. However, the problem of reordering has not been addressed much so far. The primary evalutaion metrics which exist currently for scoring translations are BLEU, METEOR, RIBES and NIST.

BLEU (Papineni et al., 2002) measures the number of overlapping n-grams in a given translation when compared to a reference translation, giving higher scores to sequential words. METEOR (Lavie and Denkowski, 2009) scores translations using alignments based on exact, stem, synonym, and paraphrase matches between words and phrases. RIBES (Isozaki et al., 2010) is based on rank correlation coefficients modified with precision. NIST (Doddington, 2002) is a variation of BLEU; where instead of treating all n-grams equally, weightage is given on how informative a particular n-gram is. We report the BLEU score as a measure to test accuracy for the 110 NMT systems to maintain brevity. However, for the language-pair English -> Hindi; we report all of the above scores. We also describe the challenges in evaluating MT accuracy keeping this language pair in consideration, however it should be noted that the same or similar challenges are faced when dealing with other language pairs as well. We use the

MT-Eval Toolkit[1] to calculate all these metrics.

It can be noted that most of the above-mentioned metrics employ some concept of word-order as well as word similarity using n-grams to score translations, which makes evaluating Hindi translations a tedious task. In addition to this, there exists a many-to-many mapping of vocabulary between English and Hindi which makes all of these scoring mechanisms less effective. For example, both translations shown in Table 2 are valid. However; since the current MT metrics rely heavily on lexical choice, there is no mechanism which takes into account the phenomena described above, which is which is quite common in Indic languages like Hindi. Hence, in addition to the metric scores, we also show sample examples with their descriptions in the following section, in order to demonstrate translation quality in a more comprehensive manner.

3 Parameters for evaluation

Since the evaluation metrics do not capture how well different linguistic phenomena are handled by our model, we perform a manual investigation and error analysis with the help of linguists. In order to have a clear insight of NMT performance as compared to SMT on various aspects, we do a side-by-side comparison of the output sentences generated by the SMT and the NMT models respectively. The linguists were asked to identify the strengths and weaknesses of NMT and SMT by ranking 200 output sentences produced by the respective models in terms of the following parameters:

- Word order
- Morphology :
  - How appropriate is the surface form selection
  - Usage of correct syntactic structures
  - Morphological agreement between words
- Phrase handling :
  - Non-translated phrases / phrases missing in the output
  - Additional phrases - Phrases occuring in the output but not in the input source sentence
- Lexical Choice - Quality and appropriateness of content words and terminology errors

[1] http://bit.ly/2p5C2FB

		Hindi	Malayalam	English
Hindi	SMT	-	10.4	27.87
	NMT	-	8.86	27.76
Malayalam	SMT	13.9	-	8.2
	NMT	12.56	-	7.88
English	SMT	26.84	5.15	-
	NMT	27.24	3.76	-

Table 3: Results of SMT and NMT on the ILCI test set

We show the results in Figure 1.

It can be observed from Figure 1 that SMT produces about twice as more errors in word order and almost thrice as more errors in syntactic and morphological structures and agreement than NMT. Thus the NMT model is able to perform significantly better than SMT for these phenomena. This results in much more fluent translations produced by the NMT model - making it a better choice in most scenarios. At the same time, the errors made in terms of lexical choice are much more in NMT than SMT. NMT also produces slightly greater number of errors in terms of missing or additional phrases. On deeper investigation, it is made clear that a majority of the lexical choice errors are due to the noise present in the training data. This leads to the insight that NMT is more prone to greater sensitivity to training noise than SMT.

To summarize, NMT performs better than SMT in most linguistic aspects, particularly in the presence of a high quality training corpus.

4 Analysis and insights

The analysis is based on the translation of prevalent sentence construction usages in the source languages. An extensive testing is done with these sentence constructions and some of the output has been reported with releveant translation and gloss in the

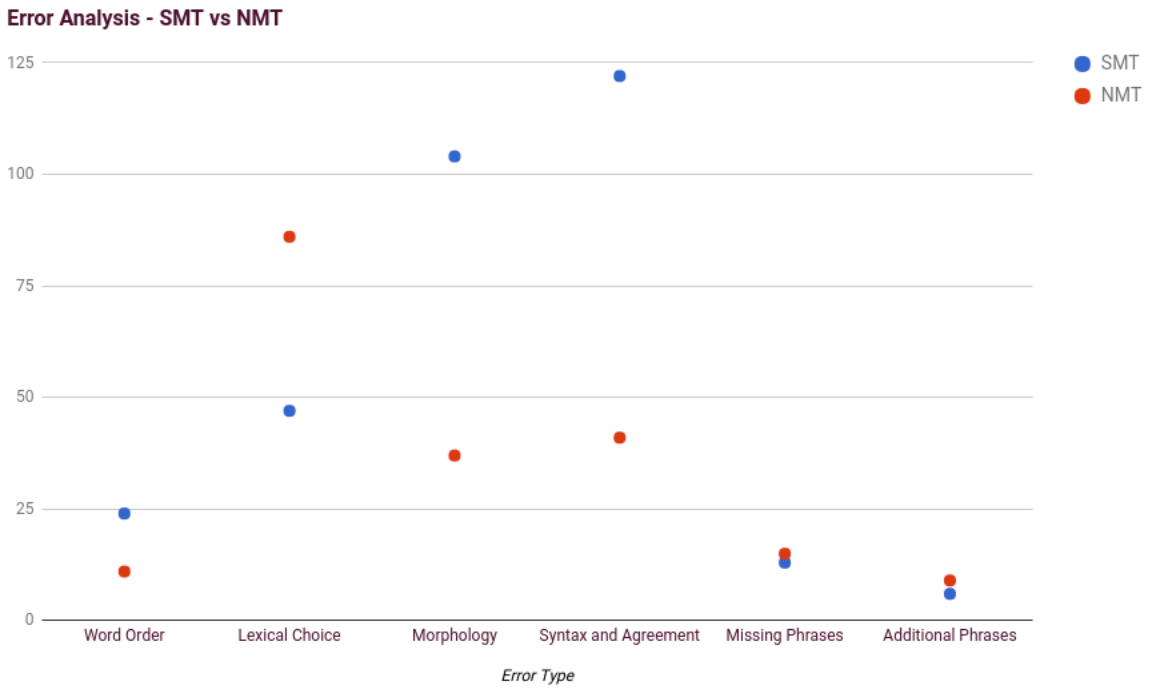


Figure 1: *Manual Error Analysis of performance of NMT with SMT*

coming sections. In order to understand the efficacy of capturing the syntactical structure of source language for the translation, we crafted simple sentences with different verbal inflections, such as transitive, intransitive, causative and different modalities in the source language. These sentences are tested and verified if the translated sentences are able to convey the same meaning from the source language. Similar attempt has been done with participle, cleft and coordination constructions in Malayalam to Hindi and English. For analysing Hindi to other languages, sentences with participle, complex predicate and coordination are tested to Malayalam and English. Tha analysis has done majorly on following sentences

- Simple sentences with different verbal inflections
- Participle, cleft and coordination constructions for Malayalam
- Participle, complex predicate, and coordination constructions for Hindi

4.1 Malayalam to English translation

SMT produces a lot of untranslated words as can be seen from the examples below, although the domain is kept the same for the manually created test set, however, the phrasing and structure is tweaked to cover all the grammatically possible constructions prevalent in Malayalam.

On the other hand, NMT shows an impressive performance in simple sentence translation from Malayalam to English. We observe that verbal inflections signalling modality is getting translated correctly in NMT(Example-3).Similarly NMT is also able to figure out variations in transitive and intransitive inflections in Malayalam to produce moderately equivalent English sentence(Example-2). At the same time NMT fails to translate imperative mood inflections correctly(Example-1). Example-1:
<SRC> വീട്ടില് കീടനാശിനി മരുന്നുകള് തളിക്കുക
<Gloss> home-LOC pesticide medicine spray-IMP.
<Translation> Pesticide sprays in home.
<SMT-ENG> വീട്ടില് കീടനാശിനി medicines തളിക്കുന്നു.

<Gloss> home-LOC pesticide medicine sprinkle-PRS.
<Translation> Pesticide medicine sprinkle in home.
<NMT-ENG> Get insecticides sprayed at home .
Example-2:
<SRC> വളരെ അധികം ദാഹം ഉണ്ടാകുന്നു
<Gloss> Very much thirsty make-PRS.
<Translation> It makes very much thirsty.
<SMT-ENG> feels very thirsty .
<NMT-ENG> One feels very thirsty .
Example-3: <SRC> ഉറങ്ങുന്ന സമയത്ത് കൊതുകു വല ഉപയോഗികണം
<Gloss> Sleep-RELAT time-DAT Mosquito net use-IMP.
<Translation> While sleeping mosquito net should be used.
<SMT-ENG> mosquito nets ഉപയോഗികണ while sleeping .
<Gloss> mosquito nets use-IMP. while sleeping .
<NMT-ENG> while sleeping mosquito net should be used.

4.1.1 Cleft constructions

Both paradigms fail to translate cleft constructions from Malayalam to English. Some of the complex syntactic constructions pertaining to the source or target language consistently fail to be learnt correctly , even though they are very common in the usage of the languages. The cleft construction could be accounted as an example as it is being used in both Malayalam and English. The SMT output is mostly erroneous and contains many untranslated words as can be seen from the following example.
Example-1:
<SRC> അനീമിക് സംബന്ധമായ രോഗങ്ങളെയാണ് വർദ്ധിപ്പിക്കുന്നത്
<Gloss> Anemic related-RELAT disease-COP increase-NOMIN.
<Translation> It is anemic relate diseases that are increased.
<SMT> anaemic വർദ്ധിപ്പിക്കുന്നു related diseases .
<Gloss> anaemic increase-PRS related diseases .
<NMT-ENG> It increases the diseases of anemic.

4.1.2 Participle constructions

The sentences with relative participle verb forms are translated incorrectly from Malayalam to English and from English to Malayalam as well. The relativised form of the verbs are predominently used in Malayalam for relative clause construction. It extends a subject sharing possibility between the realtive clause and the main clasue without the need of pronoun usage. It has also been observed that complex postpositional phrases and nomilised clauses are translated well in NMT in both directions. The example shows an erroneous traslation of a relative participle clause usage in the sentance.
Example-1:
<SRC>രോഗം പരത്തുന്ന കൊതുക് ഏഡിസ് എഡിപ്പടായ് ആണ്
<Gloss> Disease spread-RELAT mosquito aedis edippai COP.
<Translation> Disease spreading mosquito is Aedis Edippai.
<SMT-ENG> Disease and mosquito aedes aegypti .
<NMT-ENG> Malaria spreads while mosquito infected person .

4.1.3 Coordination constructions

The co-ordination constructions at clausal level are consistently translated incorrectly in both the directions in all cases of the co-ordination sample set. The construction is realised in Malayalam with complex syntactic form. The particle suffix -um is attached to all coordinating elements, but the same particle is used as an emphatic particle and also as an inclusion purpose as well. Apart from these usages the particle -um is also used for the future tense inflection. It might be the reason none of the usages of -um is translated correctly.
Example:
<SRC>ഓക്കാനം അഥവാ ഛർ ദിക്കാന് തോന്നും
<Gloss> Nausea or vomit-INF feel-FUT.
<Translation> Nausea or vomitting will feel.
<NMT-ENG> Nausea and vomiting .

4.1.4 Semantic handling in translation

A significant number of outputs generated by SMT and NMT depict correct syntactic structures but have a potent semantic loss. This is another important challenge, since the sentences being translated look like the correct usage in the target language, but the intended meaning has absolutely changed. NMT displays more such occurrences when compared to SMT, and often fails to realise the correct semantic role in the target language.

<SRC> ഇത്തരത്തിലുള്ള രോഗികളിൽ സമയത്ത് അന് റീബയോട്ടിക് ഔഷധം നല് കണം

<Gloss> This-kind-EXT-RELATE patients anti-biotic medicine give-IMP.

<Translation>This kind of patients anti-biotic medicines should be given at right time.

<SMT> Such രോഗികളിൽ time അന് റീബയോട്ടിക് medicine നല് treatment .

<Gloss> Such "patients-LOC" time "anti-biotic" medicine give-ROOT treatment .

<NMT-ENG> In such a case the medicine should be given to the doctor .

4.2 Hindi to Malayalam translation

The NMT performance on simple sentences and sentences with postpositional phrases are reasonably good, except few cases of complex syntactical co-ordination constructions and complex predicates.

Example-1:

<SRC> जिला मानसिक स्वास्थ्य कार्यक्रम के लिए उत्तराखंड के दो जिलों का चयन किया गया है।

<Translation> Two districts of Uttarakhand have been selected for District mental health programme.

<NMT> ജില്ലാ മാനസികാരോഗ്യ പരിപാടിയിൽ ഉത്തരാഖണ്ഡിലെ രണ്ട് ജില്ലകൾ തെരഞ്ഞെടുത്തു

<Gloss> District mental-health programme-LOC Uttarakhand two districts selected.

<Translation> In distirict mental-health programme two districts located in Uttarakhand selected.

<SMT> ജില്ലാ മാനസികാരോഗ്യ പരിപാടിക്കകത്ത് വേണ്ടി ഉത്തരാഖണ്ഡിലെ രണ്ടു ജില്ലകളിൽ തിരഞ്ഞടുക്കുന്നത് ചെയ്തു

<Gloss> District mental-health programme-within for Uttarakhand-LOC two districts-LOC select-NOMIN did.

<Translation> For within the district mental-health programme selecting two districts in Uttarakhand that did.

Example-2:

<SRC> टीके लेने की सलाह भी दी जाती है।

<NMT> വാക്സിൻ എടുക്കുന്നതിനുള്ള ഉപദേശം നൽകണം

<Translation> Advice should be given for taking vaccine

<Gloss> vaccine take-NOMIN-EXIST-REL advice give-IMP.

<SMT> കുത്തിവയ്പ് എടുക്കുന്നതിനുള്ള ഉപദേശം എന്നിവയും നൽകപ്പെടുന്നു

<Translation> Advice for taking Vaccination and so on are given.

<Gloss> Vaccination take-NOMIN-EXIST-RELAT advice so-on-also give-PASS-PRS.

Example-3:

<SRC> कुष्ठ रोग के लक्षणों को पहचानना बहुत कठिन है।

<NMT> കുഷ്ഠരോഗ ലക്ഷണങ്ങൾ തിരിച്ചറിയുന്നത് വളരെ അപൂർവ്വമാണ്

<Translation> Identifying leprosy syptoms that is very rare.

<Gloss> Leprosy symptoms identify-NOMIN very rare-COP.

<SMT> കുഷ്ഠ രോഗത്തിൻറെ ലക്ഷണങ്ങൾ തിരിച്ചറിയുന്നത് വളരെ കഠിനമാണ്

<Translation> Identifying leprosy's symptoms that is very hard.

<Gloss> Leprosy disease symptoms identify-NOMIN very dificult-COP.

4.2.1 Co-ordination constructions

Hindi sentences with co-ordination constructions are incorrectly translated to equivalent Malayalam sentences. This is in alignment with the previous observation that complex syntactic sentence constructions are mostly translated incorrectly. This might be due to the vast differences in the way co-ordination constructions are realised in two languages.

<SRC> माटा एनाफिलिक मच्छर स्वच्छ पानी में पैदा होता है और रात में काटता है।

<NMT> മലാരിയില് നിന്ന് വല ശേഖരിക്കരുത്

<Translation> From Malaria do not collect net.

<Gloss> Malaria-LOC from net collect-NEG.
<SMT> കൊതുക് വൃത്തിയുള്ള വെള്ളത്തിൽ ജനിക്കുകയും രാത്രിയിൽ എന്നിവ ഉണ്ടാകുന്നു
<Translation> Mosquito takes births also in fresh water night and so on makes.
<Gloss> Mosquito fresh-EXIST-RELAT water-LOC take-birth-COORD night-LOC so-on make-INTR-PRS.
<Gloss> Mosquito clean-EXIST-RELAT water-LOC take-birth-CORD night-LOC and-so-on make-INTR-PRS.

4.2.2 Complex predicate constructions

Hindi complex predicate constructions are the other set of constructions which consistently fail to be translated correctly from Hindi to Malayalam. The usage of complex predicate is confined to Hindi and the equivalent Malayalam is obtain by relative clause. The different ways of realising the semantically equal sentence might lead to problems. Secondly, scrambling of arguments with in a clause is usual in relatively free word order languages. It may also lead to the failure of this translation. The erroneous outputs are shown in the example-1 and example-2.

Example-1
<SRC> कुष्ठ रोग एक प्रकार के वायरस से होने वाला एक रोग है।
<NMT> വൈറസ് ഒരു തരത്തിലുള്ള വൈറസ് വൈറസ് ആണ്
<Translation> Virus one kind of virus is virus.
<Gloss> Virus one kind-EXIST-RELAT virus virus COP.
<SMT> കുഷ്ഠരോഗം ഒരു തരത്തിലുള്ള വൈറസ് കൊണ്ട് ഉണ്ടാകുന്ന ഒരു രോഗമാണ്
<Translation> Leprosy is due to one kind of virus making disease.
<Gloss> Leprosy one kind-EXIST-RELAT virus by make-INTR-PRS one disease-COP.

Example-2
<SRC> कुष्ठ रोग वायरस से होने वाला एक प्रकार का रोगजनन है।
<NMT> വൈറസ് വൈറസ് ബാധിച്ച ഒരു തരത്തിലുള്ള രോഗമാണ്
<Translation> Virus virus effected one kind disease is.
<Gloss> Virus virus infect-RELAT one kind-EXIST-RELAT disease-COP.
<SMT> കുഷ്ഠരോഗം വൈറസ് മൂലമുണ്ടാകുന്ന ഒരു തരത്തിലുള്ള
<Translation> Leprosy due to virus one kind of.
<Gloss> Leprosy virus due-to-make-INTR-PRS-RELAT one kind-EXIST-RELAT.

4.3 Malayalam to Hindi translation

The translations from Malayalam to Hindi using NMT do not perform better than Hindi to Malayalam. Verbal inflections are in Malayalam are not able to be translated in the apt manner in Hindi. However, certain simple sentences are handled reasonably well by NMT.

Example-1:
<SRC> നിങ്ങള് കമ്പിളി വസ്ത്രങ്ങള് ധരിക്കുക
<Gloss> you woolen clothes wear-IMP.
<Translation> You may wear woolen clothes.
<NMT-HND> आप ऊनी कपड़े पहनें ।

Example-2:
<SRC> ഉറങ്ങുന്ന സമയത്ത് കൊതുകു വല ഉപയോഗികണ
<Gloss> Sleep-RELAT time-DAT mosquito net use-IMP.
<Translation> while sleeping mosquito net should be used.
<NMT-HND> सोते समय मच्छरदानी पम्प करें ।

Example-3:
«Semantic error»>
<SRC> ഇത്തരത്തിലുള്ള രോഗികളിൽ സമയത്ത് അന് റീബയോട്ടിക് ഔഷധം നല്കണ
<Translation> This-kind-EXIST-RELAT disease-LOC time-DAT antibiotic medicine give-IMP.
<NMT-HND> ऐसे सहवास के समय एंटीबायोटिक्स औषधि देनी चाहिए ।

4.3.1 Coordination constructions

Simple nominal co-ordination constructions are successfully translated from Malayalam to Hindi. However, the complex sentential coordination is still out of its reach. The example-1 nominal coordination is translated well, whereas it failes on example-2.

Example-1:
<SRC> ഓക്കാനം അഥവാ ഛർദ്ദിക്കാന് തോന്നും
<Gloss> Nausea or vomit-INF feel-FUT.
<Translation> You will feel Nausea or vomitting.
<NMT-HND> मितली या उल्टी महसूस होना ।
Example-2:
<SRC> ഓക്കാനവും ഛർദ്ദിയും ഉണ്ടാകു
<Translation> It makes nausea or vomitting.
<NMT-HND> मतली की नियुक्ति

4.3.2 Cleft constructions

Cleft sentences in Malayalam are still a problem for Malayalam to Hindi translation. The simple sentence is translated correctly (Example 1), but its cleft form is translated incorrectly in (Example 2). This shows the dire need of an approach which can enhance NMT to learn different syntactical constructs prevalent in linguistically distant languages.
<SRC> അവന് ഡങ്കി പനി ബാധിച്ചു
<Gloss> He-DAT Dengue fever caught.
<Translation> He caught dengue fever.
<NMT-HND> जिसके कारण डेंगू बढ़ता है।
<SRC> അവനാണ് ഡങ്കി പനി ബാധിച്ചത്
<Gloss> He-COP dengue fever caught-NOMIN.
<Translation> It is him that caught fever.
<NMT-HND> वह डेंगू बुखार है।

5 Conclusion

Based on the extensive evaluation carried out on the NMT bi-directional translator with possible pairs of English, Hindi, and Malayalam, simple sentences including sentences with complex postpositional phrases are translated well in all pairs. The output quality is consistently better than SMT in most of the phenomena. An exceptional case is shown in the modal affixes of Malayalam, which are translated incorrectly to Hindi. The other important observation is that NMT is not able to decode complex verbal inflections and translate them to the target language, particularly to Hindi. A major issue of NMT is that it can not translate complex syntactic structures, particular to the source language usage. It is visible from the cleft and participle constructions of Malayalam failing to get translated to other languages and similarly complex predicate structures in Hindi to other languages. In addition to these, co-ordination constructions with conjuncts are also translated incorrectly by both SMT and NMT. These factors can serve as important guidelines to be considered when building parallel corpora for linguistically distant languages in the future, to facilitate better performance of SMT as well as NMT approaches on these language pairs.

References

Raghavachari Amritavalli. 2014. Separating tense and finiteness: anchoring in dravidian. *Natural Language & Linguistic Theory*, 32(1):283–306.

R Amritavalli. 2017. 9 nominal and interrogative complements in. *Dravidian Syntax and Universal Grammar*.

Ronald E Asher and TC Kumari. 1997. *Malayalam*. Psychology Press.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. *arXiv preprint arXiv:1608.04631*.

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1994. Anusaraka or language accessor: A short introduction. *Automatic Translation, Thiruvananthpuram, Int. school of Dravidian Linguistics*.

Akshar Bharati, Vineet Chaitanya, Rajeev Sangal, and KV Ramakrishnamacharyulu. 1995. *Natural language processing: a Paninian perspective*. Prentice-Hall of India New Delhi.

Junyoung Chung, Caglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *ICML*, pages 2067–2075.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1152–1161. Association for Computational Linguistics.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952. Association for Computational Linguistics.

KA Jayaseelan and R Amritavalli. 2005. Scrambling in the cleft construction in dravidian. *The free word order phenomenon: Its syntactic sources and diversity*, 69:137.

Karattuparambil A Jayaseelan. 2001. Ip-internal topic and focus phrases. *Studia Linguistica*, 55(1):39–75.

KA Jayaseelan. 2014. Coordination, relativization and finiteness in dravidian. *Natural Language & Linguistic Theory*, 32(1):191–211.

Girish Nath Jha. 2010. The tdil program and the indian langauge corpora intitiative (ilci). In *LREC*.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Anoop Kunchukuttan, Abhijit Mishra, Rajen Chatterjee, Ritesh Shah, and Pushpak Bhattacharyya. 2014. Sata-anuvadak: Tackling multiway translation of indian languages. *pan*, 841(54,570):4–135.

Alon Lavie and Michael J Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine translation*, 23(2):105–115.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Thomas McFadden and Sandhya Sundaresan. 2014. Finiteness in south asian languages: an introduction. *Natural Language & Linguistic Theory*, 32(1):1–27.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Interspeech*, pages 194–197.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Judith Tonhauser. 2015. Cross-linguistic temporal reference. *linguistics*, 1(1):129–154.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78:1550–1560.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1604.02201*.

Textual Relations and Topic-Projection: Issues in Text Categorization

Samir Karmakar
Jadavpur University
samir.krmkr@gmail.com

Lahari Chatterjee
Jadavpur University
lahari.chatterjee@gmail.com

Abahan Dutta
Jadavpur University
abahanjiriya@gmail.com

Abstract

Categorization of text is done on the basis of its *aboutness*. Understanding *what a text is about* often involves a subjective dimension. Developments in linguistics, however, can provide some important insights about what underlies the process of text categorization in general and topic spotting in particular. More specifically, theoretical underpinnings from formal linguistics and systemic functional linguistics may give some important insights about the way challenges can be dealt with. Under this situation, this paper seeks to present a theoretical framework which can take care of the categorization of text in terms of relational hierarchies embodied in the overall organization of the text.

1 Introduction

Multiplicity of text is the consequence of the way textual components are selected and combined into coherent wholes, apart from the factors contributing to the content. One could be suspicious about this structure-centric investigation; but it is really hard to give up structure-centricity particularly in a position when structure is crucial both in formation and representation of the text.

Distinguishibility of one text from the other depends on what sorts of textual components are selected and how are they combined into the complex structure of a text. This complex weaving of *what*s and *how*s is often termed as `textuality` - the property because of which a text attains its uniqueness. Interpretation of text, therefore, arises through the gradual unpacking of textuality. Silverman (1994) argues, "[t]he interpretation of the text brings the textuality . . . outside the text, so as to specify and determine the text in a particular fashion."

Specification and/or the determination of a text in a particular fashion possess(es) a daunting challenge to linguistics in general and computational linguistics in particular. In linguistics, this finds its way through the study of discourse, text *etc.* (Beaugrande and Dressler, 1981); whereas in computational linguistics, interest is developed due the growth of text categorization, information structure *etc.* (Nomoto and Matsumoto, 1996). A careful investigation of these two lines will reveal the fact that their respective queries and approaches are built on the question of how a text is structured: Since a structure is the embodiment of different structuring principles out of which it is made up of, explicating the process through which a text comes into being remain a central concern. Note that textuality as an account of constituents and combinatorial principles is intrinsic to the text. Therefore, the questions of specification and/or determination of a text is translated into the way the respective textuality is.

Under this situation, then, this paper is interested in understanding what textuality is. One among many ways to investigate this question is to answer how the topic of a text is projected through the characteristic but hierarchical associations of its constituents. In other words, the projection of a topic in a text in some way brings out a nexus of text-internal relations holding among constituent statements of it. If so, then, topic-spotting in one sense is an act to categorize a text with an emphasis on textuality.

If we consider topic-spotting as the single most important criterion in categorizing a text, then the paper seeks to develop an analytical account of *how the weaving of statements into a network results into the projection of its topic*. This, in turn, plays a crucial role in identifying the way a text is categorized.

Since text and textuality are inseparable from each other, our task will be of two folds: *Firstly*,

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 43–49

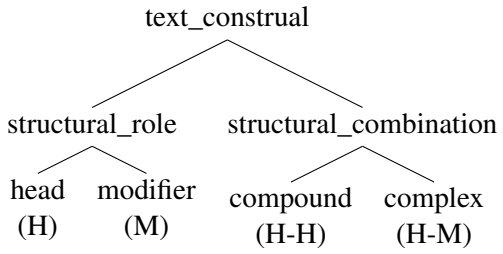
we need to come up with an analysis of the relations holding among the constituents of the text; and, *secondly*, how these relations are hierarchically organized in a text. Taken them togather, a general description of the textuality will evolve with a linguistic answer to the problem of topic-spotting.

Above mentioned two tasks will be performed over the news reports called brief. Broadsheet newspaper – generally designed in 8 columns – contains some brief news in the left most column. They are typically restricted into single column and consists of 5 to 10 sentences. The space, alloted to the briefs, is termed as Doric column for having symbolic resemblance with an archaic form of architectural order developed in Greece and Rome. A Doric column contains three to four brief news and they are ordered according to their significance which may vary from one news paper to another depending on the editorial policies. Briefs are structured.

2 Topic Projection and Textual Relations

To Taboada and Mann (2006), the communicative function of a written text is the consequence of the way words, phrases, grammatical structures and other relevant linguistic entities are getting involved into a coherent whole. In other words, a text-construal consists of certain structural roles performed by the constituents reflecting dependencies among the statements within a text and the way these roles are combined togather with the help of relations. As a result, following schematic representation of a text-construal is surfaced:

1.



As per this scheme, structural roles could be of two types namely (a) head (= H) and (b) modifier (= M). It is often noticed that head is projected as the topic. Relative saliencies of statements over each other often depends on their respective structural roles in a structural combination. For example, in a text-construal if the statements are connected with each other with the expressions - like 'and', 'or', 'but'*etc.* - chances are high for both the statements to enjoy the status of head (ex. *the moon was bright and the temperature was moderate*). This type of coordinating structure will be classified as compounded. In contrast to compounding, complex type structural combination distinguishes constituent clauses as either principle or subordinate (ex. *A guest is unwelcome when he stays too long*). Complexities involved in identifying topic of text-construal is often considered as the contribution of different statements within a network of textual relations.

In an investigation, where the topic-based categorization of text is being talked about, one need to know what is meant by topic. The topic of a text is its aboutness. More technically, topic is a statement which is entailed by the text. Following Djik (1977), we can formulate the following formal definition for the topic of a text - where text is a collection of statements in particular order.

2. A statement σ_i is the `TOPIC` of a sequence of statements $\Sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_n \rangle$ iff for each $\sigma_i \in \Sigma$ there is a subsequent Σ_k of Σ such that $\sigma_i \in \Sigma_k$ and for each successive Σ_k there exists σ_i such that $\Sigma_k \models \sigma_i$.

In other words, a statement will have the status of the topic, iff it is entailed globally by a text which it is a part of. By this, it is meant that all other local entailment relations will never have the status of topic in virtue of not being able to succeed across all subsequent collections of the statements to the ultimate text.

The most important criterion, *i.e.* the concept of entailment (represented with $\models$), for a statement to be the topic of a sequence of statements manifested through a text-construal needs some clarification: A statement *p* entails *q* when the truth of the first (*p*) guarantees the truth of the second (*q*), and the falsity of the second (*q*) guarantees the falsity of the first (*p*) (Saeed, 2009):

3.

p	$\models$	*q*
T	$\rightarrow$	T
F	$\rightarrow$	T or F
F	$\leftarrow$	F
T or F	$\leftarrow$	T

In continuation to our discussion, it could be said that a statement will have the status of topic in a text-construal, iff it is assigned to truth. As per composite truth table of entailment (3), the absolute truth of the entailed statement is subject to

the truth of a statement which is entailing the entailed. In other words, in order to have the status of topic, a statement must be in congruence with another statement in terms of the truth values. Note that the implementation of the criterion assumes a careful dissociation between *sentence* and *statement*. We will comment on this issue in our discussion below.

In a text, the other statements, acting as entailers, are used in modifying the entailed sentence (i.e. the topic) in various capacities - say for example elaborating, extending, enhancing etc. We will call them `relators`. It is not always necessary for a relator to be expressed explicitly in a text construal. What is worth mentioning is the fact that a statement with the status of topic is modified by modifiers in a structured manner. Having said this, it is emphasized that topic of a text hierarchically organizes various other functions with which it is associated. As an example consider the following piece from a brief published in Anandabazar on July 31, 2017:

4. (a) jhARkhaNDe bhArI briSTir AshangkA
Jharkhand.loc heavy rain.poss fear
nA thAkAy Dibhisir jal chArA
not having DVC.poss water release
kAmeche
reduce.perf.pres.3
Having no fear of heavy rain in Jharkhand, water release of DVC has reduced

(b) rAjya prashAsan tAi trAner
State administration therefore relief.poss
kAje bARti najar dicche
work-loc extra attention give.impf.pres.3
State administration, therefore, is giving more attention to the relief work.

Note that (4a) and (4b) both of them separately have their own topics: When (4a) is about the reduction in the release of water, (4b) is about paying extra attention to the flood situation. However, when these two statements are put togather in a text, the topic or aboutness of the resultant text is determined by the characteristic relation holding between (4a) and (4b). A careful look into (4) will reveal the fact that the topic of it is (4a). Being a topic, (4a) will enjoy the status of head (H) and (4b), the status of modifier (M). But what type of relation (4a) and (4b) are in? We can define the relation as `therefore`, because (4a) is reporting a situation that logically results into the situation reported in (4a). Moreover, (4b) contains an explicit lexical item *tai* in it to show how (4b) is modifying (4a). Further, reporting of (4a) is more central to reporter's purpose in putting forth the H-M combination than the reporting of (4b). Thus the relation could be termed as `therefore` - which licenses complex strucural combination. This discussion can be summarized in the following way:

5. $\texttt{therefore}(4a, 4b) \models 4a$
interpreted as ((4a), `therefore` (4b)) entails (4a) with a reference to a text-construal

To distinguish a statement from a sentence, we will introduce Greek alphabet σ with the provisions of using subscripts. Later on, it would be shown that a single sentence can have more than one statements in it. For this time, lets consider the statement expressed in (4a) is σ_i and the statement expressed in (4b) is σ_j. As a result, (5) will be converted into

6. $\texttt{therefore}(\sigma_i, \sigma_j) \models \sigma_i$
where `therefore` is a relator connecting a head statement with its modifier statement with a reference to a sequence of statements corresponding to a text-construal

The local relation(s) holding between two statements are getting modified when a third sentence is added with it. Consider the following sentence as the part of (4):

4. (c) rabibAr goghATe jal nAmleo
Sunday Goghat.loc water decreasing.prt
nadIgulir jalastar beshi thAkAy ghATAl
river.pl.poss water more having Ghatal
o khAnAkule teman nAmeni
and Khanakul.loc such decreasing.neg
On Sunday, in Goghat, though the water level decreased, in Ghatal and Khanakul no such change is noticed due to the high water level in the rivers.

Inclusion of (4c) will have its impact on the existing relational pattern because of effecting the distribution of roles and their combinatorial pattern in the resultant text-construal: (4c) represents a statement which is contradictory to the text-construal comprised of (4a) and (4b). This time no relation is explicitly mentioned. We will name this relation `contrarily` - since (4c) is providing an information which is contradicting with the previously stated information. Because of being compound type structural combination, resultant text-

construal will entail both (4a) and (4c): In compound type structural combination both the statements have the status of heads. As a consequence both (5) and (6) will be augmented or modified in the following ways:

7. $\texttt{contrarily}(\texttt{therefore}(4a, 4b), 4c)$
$\models \begin{cases} 4a \\ 4c \end{cases}$
Interpreted as,
(((4a) `therefore` (4b)), contrarily (4c)) entails 4a and 4c with respect to a text-construal

Conversion to the relational scheme of corresponding statements will give us the following result:

8. $\texttt{contrarily}(\texttt{therefore}(\sigma_i, \sigma_j), \sigma_k)$
$\models \begin{cases} \sigma_i \\ \sigma_k \end{cases}$ with respect to a sequence of statements corresponding to the text-construal

Note that the entailment relation is changed with the addition of newer statement. If this is a deviation from what is claimed in (2), then one should have some satisfactory answer to the question of how topics are licensed to percolate from one text-construal to its successive text-construals. No doubt, the answer to this problem has to come from the characteristic interactions holding between the structural aspect (= syntactic) and the meaning aspect (= semantic) of a text-construal. By strucural aspect, different combinations of H(ead) and M(odifier) are meant; whereas the meaning aspect is primarily concerned about the topic as well as entailment relations. The proposed solution to this problem will be explained in Section 4.

2.1 Sentence Internally Topic Projection

Though we are concerned about the topic spotting with a focus on the sequences of statements primarily at the sentential level, it is possible to trace back the topic from the sub-sentential level analysis - because a sentence can contain more than one statements. Therefore, to trace back our analysis from the subsentential level, we need to identify the subsentential constituents in the following way:

4(a) $\sigma_{i.1}$: jhARkhaNDe bhArI briSTir aAshangkA
Jharkhand.loc heavy rain.poss fear
nA thAkAy
not having
Having no fear of heavy rain in Jharkhand,

$\sigma_{i.2}$: Dibhisir jal chArA kameche
DVC.poss water release reduce.perf.pres.3
water release of DVC has reduced.

4(b) σ_j: rAjya prashAsan tAi trAner
State administration therefore relief.poss
kAje bARti najar dicche
work-loc extra attention give.impf.pres.3
State administration, therefore, is giving more attention to the relief work.

4(c) $\sigma_{k.1}$: rabibAr goghATe jal nAmleo
Sunday Goghat.loc water decreasing.prt
On Sunday, in Goghat, though the water level decreased

$\sigma_{k.2}$: nadIgulir jalastar beshi thAkAy
river.pl.poss water-level more having
having high water level in the rivers

$\sigma_{k.3}$: ghATAl o khAnAkule teman
Ghatal and Khanakul.loc such
nAmeni
decreasing.neg
in Ghatal and Khanakul no such change is noticed

Now, consider the case of (4a) which is a collection of following two statements: (i) there is no fear of heavy rain (= $\sigma_{i.1}$), and (ii) DVC is reducing the water release (= $\sigma_{i.2}$). Here in this case the former one is the modifier and the latter one is the head. First one is the reason for the second one. Alternatively, we can say, second one is the consequence of the first one:

9. $\texttt{consequently}(\sigma_{i.1}, \sigma_{i.2}) \models \sigma_{i.2}$

Similarly, (4c) as a complex sentence is made up of three distinct statements: (i) decreasing of the water level in Goghat region on Sunday ($\sigma_{k.1}$), (ii) having more water in the rivers (= $\sigma_{k.2}$), and (iii) not decreasing water levels in Ghatal and Khanakul regions (= $\sigma_{k.3}$). Here, (iii) is the head modified with (ii).

10. $\texttt{consequently}(\sigma_{k.2}, \sigma_{k.3}) \models \sigma_{k.3}$

Being in contrast with topic-projection of (10), the statement $\sigma_{k.1}$ will also have the status of head. As a result, along with $\sigma_{k.3}$, $\sigma_{k.1}$ will also be entailed by the resultant sequence of statements:

11. $\texttt{contrarily}(\sigma_{k.1}, \texttt{consequently}(\sigma_{k.2}, \sigma_{k.3}))$
$\models \begin{cases} \sigma_{k.1} \\ \sigma_{k.3} \end{cases}$

Later on, in Section 4, the rest of this story of topic projection will be presented. Here, in this point of our discussion, we would rather like to turn towards the questions of how a particular relation existing between two statements are identified, and how a statement is assigned to the topic. To address these issues, in Section 3, the underlying conceptual framework for topic extraction is explained.

3 Conceptual Framework for Topic Extraction

The process involved in the categorization of text in terms of its topic extraction, as is described in Section 2, can be conceptualized in the following way: As per our understanding, any text (like, brief) can be conceived as the sequence of statements. Each of these statements in isolation has a topic - no matter, how trivial it may sound. As a part of a text-construal, each one of them is related with some other statement. As is discussed earlier, (i) either one of the each pair has the status `head` and the other is the `modifier` (as in subordination), or (ii) both of them are of head status (as in coordination). Relator along with the concept of structural saliencies plays a crucial role in determining the topic of the text-construal made up of the constituent statements. Formally, a relator can be defined as,

12. $\mathfrak{R} : \mathfrak{S} \times \mathfrak{T}$ where $\mathfrak{S}$ is the set of statements and $\mathfrak{T}$ is the set of topics;

Assignment of a topic τ_i to a statement σ_i is a subjective task. This subjective dimension can be discussed in terms of a characteristic function:

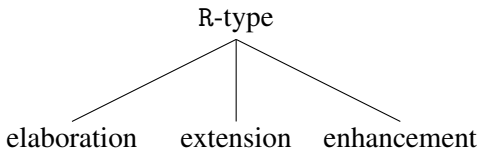
13. $\mathfrak{f} : \mathfrak{R} \rightarrow \{1, 0\}$

Before getting into our final section, lets have a look into the issues of relations.

3.1 Textual Relations

What seems to be central to the conceptual framework discussed in Section (3) is an account of different types of textual relations - for which we have relators of both implicit and explicit types. Few such relations - like `therefore`, `contrarily`, and `consequently` - are already mentioned in our discussion. Relations are important for the formation of complex statements. More specifically, relators are crucial in maintaining the coherence of a text. Three major types of relators will be discussed in section following the proposal of Systemic Functional Linguistics (Eggins, (Eggins, 2004)):

14.



These three types of relations are crucial in explaining the way text-construal incorporates the newer forms of information through the gradual increment of the relational network. While *elaborating* we are restating a statement for better clarity. In case of *extending*, additional statements are supplied; and the act of *enhancing* is used to indicate the further development of the meaning already communicated by a statement. Being the part of the typological classification of the relators, each of them are defined set-theoretically over a set of relators. In (15), these three types of relators are represented with their respective members:

15. (a) elaboration
`clarify`, `restate`, `exemplify`, `instantiate`, `illustrate`, `in_other_words`, `to_be_more_precise`, `as_a_matter_of_fact`, `actually`, `in_fact` etc.
(b) extension
`and`, `but`, `additionally`, `furthermore`, `moreover`, `excepting`, `apart_from_that`, `alternatively`, `on_the_other_hand`, `on_the_contrary`, `instead` etc.
(c) enhancement
`after`, `before`, `next`, `then`, `therefore`, `simultaneously`, `sequentially`, `until`, `since`, `now`, `similarly`, `yet`, `still`, `despite`, `though`, `consequently` etc.

The algebraic system underneath the process of topic sorting as a most important criterion to categorize a text, then, is a tuple $\langle \mathfrak{S}, \mathfrak{T}, \mathfrak{R}, \mathfrak{f} \rangle$ - which consists of a non-empty set of statements, a set of topics, a class of relations defined over the cartesian product of $\mathfrak{S}$ and $\mathfrak{T}$, and a characteristic function assigning each of the relations to

the set of values. In addition to the issues discussed above, what seems to be most crucial for the relations mentioned in (15) is their classification either as compound or complex type relations. More specifically, it is important to know which type of relations permit coordinate structures and which ones permit subordinate structures. From a gross observation, it seems the relations categorized as elaboration type and enhancement type permits subordination; whereas extension type relations are useful in constructing coordination.

4 Discussion

This section is dedicated to the hierarchical organization of the relations in a text-construal. This is done with the help of attribute value matrix. As per the conventions, set earlier, H and M are used to mark the organizational roles associated with the statements. Functional dependencies existing between the statements, along with their respective entailments, are represented with the aid of the relations. Apart from these, T and τ are used as subscripts to mark the status of the statements as topic. A statement marked with T is likely to percolate as topic to the next stage of construal as against the statement with subscipted τ confined within the text-construal which it is a part of. Note the identification of a topic (τ) either as a head (H) or as a modifier (M) is determined solely by the semantics of the relations. The topic is marked as T when it is identified as H. In addition to this, a concept of *rank* needs to be introduced here: If a matrix embeds another matrix in it, then the former one would be considered as of higher rank construal in comparison to the latter one. Consider the following examples:

16. matrix with higher rank with respect to (17):

$$\begin{bmatrix} \mathrm{H}_{\boxed{T}}{:}\texttt{consequently}(\sigma_{i.1},\sigma_{i.2}) \models \sigma_{i.2} & \begin{bmatrix}\mathrm{M}_{\boxed{\tau}}{:}\sigma_{i.1} \models \sigma_{i.1}\end{bmatrix} \\ & \begin{bmatrix}\mathrm{H}_{\boxed{T}}{:}\sigma_{i.2} \models \sigma_{i.2}\end{bmatrix} \end{bmatrix}$$

(16) is considered as a matrix of higher rank with respect to the matrices enumerated in (17):

17. matrices with lower ranks

(a) $\begin{bmatrix}\mathrm{M}_{\boxed{\tau}}{:}\sigma_{i.1} \models \sigma_{i.1}\end{bmatrix}$ (b) $\begin{bmatrix}\mathrm{H}_{\boxed{T}}{:}\sigma_{i.2} \models \sigma_{i.2}\end{bmatrix}$

However, (16) will be considered as a matrix with lower rank with respect to (18).

18. Topic percolation through the network of textual relations:

$$\begin{bmatrix} \Sigma & \boxed{1}\begin{bmatrix} \mathrm{H}_{\boxed{T}}{:}\mathfrak{t}(\sigma_i,\sigma_j) \models \sigma_i & \begin{bmatrix}\mathrm{H}_{\boxed{T}}{:}\mathfrak{c}(\sigma_{i.1},\sigma_{i.2}) \models \sigma_{i.2} & \begin{bmatrix}\mathrm{M}_{\boxed{\tau}}{:}\sigma_{i.1} \models \sigma_{i.1}\end{bmatrix} \\ & \begin{bmatrix}\mathrm{H}_{\boxed{T}}{:}\sigma_{i.2} \models \sigma_{i.2}\end{bmatrix}\end{bmatrix} \\ & \begin{bmatrix}\mathrm{M}_{\boxed{\tau}}{:}\sigma_j \models \sigma_j\end{bmatrix} \end{bmatrix} \\ & \boxed{2}\begin{bmatrix} \mathrm{H}_{\boxed{T}}{:}\mathfrak{C}(\sigma_{k.1},\mathfrak{c}(\sigma_{k.2},\sigma_{k.3})) \models \begin{matrix}\sigma_{k.1}\\ \sigma_{k.3}\end{matrix} & \boxed{3}\begin{bmatrix}\mathrm{H}_{\boxed{T}}{:}\sigma_{k.1} \models \sigma_{k.1}\end{bmatrix} \\ & \boxed{4}\begin{bmatrix}\mathrm{H}_{\boxed{T}}{:}\mathfrak{c}(\sigma_{k.2},\sigma_{k.3}) \models \sigma_{k.3} & \begin{bmatrix}\mathrm{M}_{\boxed{\tau}}{:}\sigma_{k.2} \models \sigma_{k.2}\end{bmatrix} \\ & \begin{bmatrix}\mathrm{H}_{\boxed{T}}{:}\sigma_{k.3} \models \sigma_{k.3}\end{bmatrix}\end{bmatrix} \end{bmatrix} \end{bmatrix}$$

For the sake of the brevity and the ease of the presentation, in (18), we have used following abbreviations: $\texttt{therefore} = \mathfrak{t}$, $\texttt{consequently} = \mathfrak{c}$, $\texttt{contrarily} = \mathfrak{C}$.

Needless to say, matrices mentioned in (17) would be of *lowest* rank because of not embedding any other matrix; on the other hand, (18) will be of *greatest* rank in virtue of not being embedded in other matrix. Significance of relative ranks is useful in explaining how the projection of topic is taking place within the text-construal.

As per attribute value matrix of (18), the topic of a sequence of statements, Σ corresponding to a text construal (4), will be that statement which is embedded in all successive matrices of higher ranks as head (= H). In other words, topic of a modifier is not licensed to be the topic of a matrix with higher rank within which the modifier is embedded. As per this assertion, then, it is not hard to argue why the topics of the lowest rank text-construals as modifier fail to percolate as a topic in the text-construals which are in immediately higher ranks.

As the analytical framework outlined and discussed above, the text mentioned in (4) have two distinct topics:

19. $\Sigma \models \begin{cases} topic_{4a-b}\ marked\ as\ 1 \\ topic_{4c}\ marked\ as\ 2 \end{cases}$

These two topics are projected by two different text-construals which are in equal rank as is obvious from the attribute-value matrix of (18). Similar situation prevails in case of the matrices marked with 3 and 4. If two topics are of equal rank, then chances are high for the respective texts to be independent of each other. In other words, two topics with equal rank are combined togather into a text-construal with the aid of those relations which are crucial in producing compounded structures.

On the basis of this discussion, we can argue that Dijk's criterion (1977) for topic identification

mentioned in (2) seeks the following modification: Any text as a sequence of statements will have one and only one topic iff the constituent textual relations are in complex type structural combination. Compounding of constituent relations will indicate their respective projections as independent of each other. In such a situation, the text-construal can be broken into two independently occurring texts. This could be used as a potential clue to the auto-editing of briefs in particular and news reporting in general.

References

R. A. De Beaugrande and W. U. Dressler 1981. *Introduction To Text Linguistics* Longman, New York.

M. Taboada and W. C. Mann 2006. Rhetorical Structure Theory: Looking Back and Moving Ahead *Discourse Studies*, 8(3), pp. 423-459.

T. A. van Djik 1977. *Text and Context: Explorations in the semantics and pragmatics of discourse* Longman, London.

J.I. Saeed 2009 *Semantics* (3rd edition) Wiley-Blackwell, Oxford.

H. J. Silverman. 1994. *Textualities: Between Hermeneutics and Deconstruction* Routledge, London.

T. Nomoto and Y. Matsumoto 1996 Exploiting Text Structure For Topic Identification *Workshop On Very Large Corpora*, 101-112

S. Eggins 2004 *An Introduction to Systemic Functional Linguistics* Second Edition Continuum, New York

POS Tagging For Resource Poor Indian Languages Through Feature Projection

Pruthwik Mishra[1] **Vandan Mujadia**[2] **Dipti Misra Sharma**[3]
Language Technologies Research Center, IIIT Hyderabad
Kohli Center On Intelligent Systems
pruthwik.mishra@research.iiit.ac.in,vmujadia@gmail.com,dipti@iiit.ac.in

Abstract

We present an approach for POS tagging with out any labeled data. Our method requires translated sentences from a pair of languages. We used feature transfer from a resource rich language to resource poor languages. Across 8 different Indian Languages, we achieved encouraging accuracies without any knowledge of the target language and any human annotation. This will help us in creating annotated corpora for resource poor Indian languages.

Keywords

POS, NLP, corpus, parallel corpora, Feature Transfer, Alignment, Mapping

1 Introduction

Part-Of-Speech(POS) (Bharati et al., 2007) Tagging is considered as a preliminary task for various Natural Language Processing(NLP) tasks. POS Tagging primarily assigns class labels to words based on some extracted features. The POS tagged corpus can further be used for parsing, building lexical dictionaries, frequency lists and many more subsequent tasks [1]. For automatic POS tagging, the state-of-the-art POS taggers use large POS annotated data sets and try to learn the appropriate class labels for words depending on various hand annotated features. There are many Indian Languages which are unexplored due to the unavailability of annotated corpora. But recently, there has been a lot of efforts to create monolingual as well as bilingual corpora for different Indian Languages.

Hindi is resource rich in this regard as there are many linguistic resources created for Hindi. One of the notable corpus available is the Hindi Treebank (Bharati et al., 2006). Statistical POS taggers trained on Hindi Treebank data for Hindi achieved around 93% accuracy (Gadde and Yeleti, 2008). Stochastic or Statistical Taggers are also used for Indian languages like Punjabi, Urdu, Marathi, Telugu, but their accuracies fall due to lack of large annotated corpora. POS Taggers for other Indian Languages have not been evaluated. So these languages are resource poor in terms of high quality linguistic annotated data.

The motivation behind this work is to create lexical resources for resource poor Indian languages. All the Indian Languages are morphologically rich, prefixes and the word ending suffixes encode a lot information about the category of the word. We try to leverage these similarities and availability of Hindi corpus for creating resources for other languages.

The paper is divided as per the following. In the section 2, we describe Background of Projection using aligned Corpora. Section 3 gives an account of Corpus Details, section 4 describes the Algorithm and various tools used. Section 5 presents the Experimental Results and in the subsequent section 6, we have the error analysis. The future work is discussed in the concluding section.

2 Background

Many supervised learning techniques reported state-of-the-art accuracy of around 90% for POS tagging in Indian Languages. POS Tagging is more accurate in most of the Indo-Aryan Languages while the results are poorer

[1] http://www.ahds.ac.uk/creating/guides/linguistic-corpora/

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 50–55

for agglutinative Dravidian Languages. But one major bottleneck in POS tagging is the requirement of a large labeled corpus which is difficult to create. To overcome this difficulty, many researchers have employed unsupervised techniques which are less accurate (Accuracies reported in the range of 70-80%) (Christodoulopoulos et al., 2010)

So in our approach, we leveraged the gold quality corpus of a resource rich language and transferred features to a resource poor language. The only resource available to us is a parallel corpus with the resource rich one. We report the results using two different tag-sets - one being the tag-set defined for Indian Languages named as the IIIT - tagset (Bharati et al., 2006) which is fine-grained and the Universal Tag-set (Petrov et al., 2011) mostly used for Unsupervised and semi-supervised POS-Tagging which is coarse-grained. We evaluated on 8 Indian Languages and obtained overall average accuracies of 81%.

(Yarowsky et al., 2001) introduced robust projections across aligned corpora. They used a statistical POS tagger for tagging source side text and transferred the POS tags to the target side from the word alignments obtained. The noisy transfers were filtered out re-estimating the the most frequent tag sequence model. Other works (Das and Petrov, 2011) used bilingual projections using Universal Tag-set. All these methods employ Label Propagation (LP) to transfer the tags from labeled data to unlabeled data. These are examples of semi-supervised techniques and major work has been done on European languages. The work of (Das and Petrov, 2011) is closest to (Yarowsky et al., 2001). These methods are evaluated on data sets which are very similar to the parallel corpus. Direct transfer of tags using raw projection can lead to very noisy POS tags. Instead of directly matching words from the word alignments available; we use the feature of the words which are clear indicators of POS tags realized through rich morphology. Because of the limited size of bi-text (parallel corpus) chances of finding exact matching words gets reduced. We do not use the observed word as a feature. For avoiding non-matched features, we use back-off smoothing. Thus we have an approximate feature representation for any word occurring in the corpus. The features used suffice to the back-off model. The Suffixes and prefixes provide valuable cue in the identification of a particular POS category. Additionally, suffixes help to disambiguate between various similar categories.

We trained the models on general domain and tested on health domain data. The performance of our models is comparable to the state-of-the-art systems in out-of-domain data.

Language	Domain	#Tokens
Hindi	Health	368K
Hindi	Tourism	474K
Marathi	Health	382K
Marathi	Tourism	278K
Konkani	Health	346K
Konkani	Tourism	328K
Urdu	Health	371K
Urdu	Tourism	473K
Bengali	Health	300K
Bengali	Tourism	387K
Gujarati	Health	329K
Gujarati	Tourism	388K
Punjabi	Health	386K
Punjabi	Tourism	425K
Tamil	Health	313K
Tamil	Tourism	312K
Telugu	Health	316K
Telugu	Tourism	316K
Malayalam	Health	286K
Malayalam	Tourism	291K

Table 1: ILCI Corpus Details

3 Corpus Details

We used two data sets for our experiments.

1. ILCI (Indian Languages Corpora Initiative) parallel corpora
2. Hindi Tree-bank

The data used for parallel corpora was the ILCI (Choudhary and Jha, 2014) corpora released for different languages. We have experimented on 8 languages :- Punjabi, Konkani, Bengali, Telugu, Malayalam, Urdu, Marathi, Gujarati

The details of the ICLI corpus are shown in Table 1, the number of sentences in each

Data-Set	#Sentences	#Tokens
Hindi Treebank	21K	450K

Table 2: Hindi Treebank Details

language was 25K.

The Hindi Treebank (Bharati et al., 2006) creation task was taken up at IIIT, Hyderabad. The treebank is a multi-layered representation of sentences with syntactic and semantic annotation. The syntactic annotation includes morph analysis, POS Tagging, Chunking of words or tokens occurring in a sentence. We used Hindi treebank for ensuring high quality projection of POS tags. The Hindi Treebank is annotated with POS tags from IIIT tag-set(tagging annotation guidelines described in (Bharati et al., 2006)). The details of the Hindi Treebank is presented in Table 2. We also converted IIIT tags to Universal tags (Petrov et al., 2011) and evaluated the POS Tagging accuracy for both the tagsets. Universal Tag-set is often used for projection techniques to remove ambiguities related to finer grained tags.

4 Algorithm

Our approach is an example of feature representation transfer. We transfered the knowledge acquired from a language to another language. In this paper, the source language used for feature transfer was Hindi and the target languages for projection were resource poor Indian Languages explained in the above section. The scarcity of data for any language will not impact the performance if a huge training data set is available for another language. With a mapping between the feature sets of the concerned languages, we have the luxury of creating training data of comparable size for a resource poor language.

Our algorithm has 5 steps.

4.1 Word Alignment

Learning word alignments from the parallel text is the first step in our approach. We used GIZA++ tool [2] for capturing the word level alignment between sentences that are aligned for a pair of languages. The raw text files for a source language and target language serve as the inputs for the tool. In this case, the source language was Hindi and target language was any of the resource poor Indian languages. GIZA++ tool finds the alignments between words with translation probabilities. It also generates files with translation probabilities of aligned sentences. A word can have multiple alignments, but we selected the alignment with highest probability. We were able to eliminate noisy alignments by only selecting the most likely alignment for a word.

[2]https://github.com/moses-smt/giza-pp.git

Feature	Example
Prefix length 1	प
Prefix length 2	पत
Prefix length 3	पत्
Prefix length 4	पत्र
Prefix length 5	पत्रक
Prefix length 6	पत्रका
Prefix length 7	पत्रकार
Suffix length 1	◌ं
Suffix length 2	◌ों
Suffix length 3	रों
Suffix length 4	ारों

Table 3: Features for Hindi

4.2 Feature Selection

As POS Tagging is sequence labeling task, certain features need to be captured in classifying the words and assigning them appropriate tags. Indian Languages are morphologically rich, therefore prefixes and suffixes provide a lot of information about the category of the word. For Indian Languages, we considered the following morph features:-

- The prefix characters up to 7 characters
- The suffix characters up to 4 characters
- Length of the word
- Context Window size of 3 (Previous word, Current word and Next word)

For example the feature representation for a Hindi word पत्रकारों (Patrakāron - Journalists) is given in Table 1:

The above features are extracted from the words present in the Hindi Treebank. After this step, all the words in the Hindi Treebank are represented in terms of their features.

if LENGTH($word$) < LENGTH($feature$) **then**
▷ The prefix length can vary from 1-7 and suffix length can range from 1-4
$feature \leftarrow NULL$

Feature	Source	Mapped
Prefix 1	व	ਵ
Prefix 2	वि	ਵਿ
Prefix 3	विव	ਵਿਆ
Prefix 4	विवा	ਵਿਆਹ
Prefix 5	विवाह	ਵਿਆਹੀ
Prefix 6	विवाहि	ਵਿਆਹੀਆ
Prefix 7	विवाहित	ਵਿਆਹੀਆਂ
Suffix 1	त	◌ੰ
Suffix 2	◌ित	ਆਂ
Suffix 3	हित	◌ੀਆਂ
SUffix 4	◌ाहित	ਹੀਆਂ

Table 4: Feature Mapping Between Hindi and Punjabi

4.3 Mapping File Creation

After obtaining word level alignments, we created feature level mapping files based on the features defined in the previous subsection. As the word alignments with highest probabilities were taken into account, the corresponding feature files supported the best possible mapping from the source language to the target language. The example of Hindi - Punjabi pair mapping file is given in Table 4. The word in Hindi is 'विवाहित' (Vivāhita - married) and the aligned word in Punjabi is 'ਵਿਆਹੀਆਂ' (Vivāhita - married). We did not normalization the text before the feature transfer. This was done keeping in our effort of not including any language specific information for any resource poor language. If the same feature got mapped to multiple target features, we selected the target feature with highest probability. There are 7 features corresponding to prefixes and 4 features for suffix, so there are total 11 feature mapping files from Hindi to one of the resource poor languages.

4.4 Feature Transfer

This is the most vital step of the algorithm in which the features obtained from the tokens present in the Hindi Treebank are transferred to other languages. Hindi Treebank data is used to avoid noisy projections from Hindi to resource poor languages. Each word in the treebank is represented by the features described above. From the mapping files, we obtain a particular Hindi feature and its corresponding mapped feature in the language under test. If a feature is missing from the feature mapping file, a back-off model finds the next lower length feature. These feature transformations are essential for a sound representation in the target side. Otherwise, in case of features that are not-found, the target side would have been filled with NULL features and will affect the performance of the POS Tagger.

4.5 Creation of Training Model

This step is the training phase of POS Tagging where the samples are assigned pre-defined labels i.e. POS tags. We have used Conditional Random Fields(CRF) classification algorithm for our task. The conditional random fields are implemented via CRF++ [3]tool. This tool receives features in the form of a template. From the training samples, CRF creates a model assigning feature weights to the individual features. After the model is created, a test data set is used to predict the POS tags of the test samples.

The whole process can be modeled as a composition of different functions. The final model for predicting POS tags is given in equation 1

$$final\_model = M(T(f(A(x, y)))) \quad (1)$$

where A→ Alignment between word x and y
f→ Feature Representation for the words x, y
T→ Transfer of Features from x to y
M→ Model creation using the transferred features

5 Experiments & Results

The experimental results are shown in Table 5. The 1st entry for Gujarati corresponds to the word alignments obtained from Original Hindi - Gujarati ILCI parallel corpus. The other entry reflects the accuracy obtained after preprocessing the parallel corpus. Case markers are not present as separate tokens in Gujarati. As a result of this, there were no alignments for many post-positions used as case markers

[3]https://taku910.github.io/crfpp/

Language	Accuracy	#Train	#Test
Gujarati	80.5	450K	6.5K
Gujarati	86.2	380K	6.5K
Punjabi	84.3	450K	9K
Urdu	85.6	450K	7.5K
Konkani	76.9	380K	7.5K
Bengali	75.5	380K	7.5K
Telugu	74.2	380K	7K
Marathi	77.7	380K	7K
Malayalam	65.01	380K	6K

Table 5: Accuracies for Different Languages

in Hindi. So we combined the post-positions with the preceding noun on the Hindi side and found word alignments from the changed parallel corpus.

6 Error Analysis

Languages which are close to Hindi gave better results than other languages. Gujarati, Urdu and Punjabi have similar syntactic structure to Hindi with minor variations. Gujarati does not separate case markers as Hindi, where the case markers are present as a suffix in the nouns. Other languages like Bengali, Konkani, Marathi, Telugu, Malayalam are morphologically richer than Hindi. So the word level alignment is less accurate. To overcome this shortcomings, we combined the post positions marking the case, the auxiliary verbs with the preceding head categories. The head category in the former was the noun and the corresponding head in the latter was the verb. By incorporating these changes, we were able to capture the inherent syntactic behavior of these languages. The increase in accuracy for Gujarati showed this. This heuristic helped to create a better feature mapping (Petrov et al., 2011).

The sources of major errors are listed as follows:

- Errors in the Gold Data
- Ambiguities between Nouns and Adjective
- Ambiguities between particle and conjunction
- Ambiguities between Verbs and Auxiliary Verbs
- Alignment Issues

7 Future Work

We would extend this work to other resource poor Indian languages. Semantic Clustering using neural networks is an interesting area to explore. As the efficiency of the system relies heavily on the word alignments between a pair of languages, new methods can be implemented to improve the alignments. We will experiment with other sequence labelers like structured perceptron (Collins, 2002), SVM-Struct (Tsochantaridis et al., 2004), sequence-to-sequence learners(Sutskever et al., 2014). We will explore (Das and Petrov, 2011) the label propagation of tags between languages.

Acknowledgement

We thank DIT for providing ILCI corpus which enabled us to carry out the research. We would also like to thank people of different languages for helping out in the error analysis of POS tagging.

References

Akshar Bharati, Rajeev Sangal, Dipti Misra Sharma, and Lakshmi Bai. 2006. Anncorra: Annotating corpora guidelines for pos and chunk annotation for indian languages. *LTRC-TR31*.

Akshar Bharati, Rajeev Sangal, and Dipti M Sharma. 2007. Ssf: Shakti standard format guide. *Language Technologies Research Centre, International Institute of Information Technology, Hyderabad, India*, pages 1–25.

Narayan Choudhary and Girish Nath Jha. 2014. Creating multilingual parallel corpora in indian languages. In *Human Language Technology Challenges for Computer Science and Linguistics*, pages 527–537. Springer.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised pos induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 600–609. Association for Computational Linguistics.

Phani Gadde and Meher Vijay Yeleti. 2008. Improving statistical pos tagging using linguistic feature for hindi and telugu. *Proc. of ICON*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, pages 1–8. Association for Computational Linguistics.

An Exploration of Word Embedding Initialization in Deep-Learning Tasks

Tom Kocmi and **Ondřej Bojar**
Charles University,
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
surname@ufal.mff.cuni.cz

Abstract

Word embeddings are the interface between the world of discrete units of text processing and the continuous, differentiable world of neural networks. In this work, we examine various random and pretrained initialization methods for embeddings used in deep networks and their effect on the performance on four NLP tasks with both recurrent and convolutional architectures. We confirm that pretrained embeddings are a little better than random initialization, especially considering the speed of learning. On the other hand, we do not see any significant difference between various methods of random initialization, as long as the variance is kept reasonably low. High-variance initialization prevents the network to use the space of embeddings and forces it to use other free parameters to accomplish the task. We support this hypothesis by observing the performance in learning lexical relations and by the fact that the network can learn to perform reasonably in its task even with fixed random embeddings.

1 Introduction

Embeddings or lookup tables (Bengio et al., 2003) are used for units of different granularity, from characters (Lee et al., 2016) to subword units (Sennrich et al., 2016; Wu et al., 2016) up to words. In this paper, we focus solely on word embeddings (embeddings attached to individual token types in the text). In their highly dimensional vector space, word embeddings are capable of representing many aspects of similarities between words: semantic relations or morphological properties (Mikolov et al., 2013; Kocmi and Bojar, 2016) in one language or cross-lingually (Luong et al., 2015).

Embeddings are trained *for a task*. In other words, the vectors that embeddings assign to each word type are almost never provided manually but always discovered automatically in a neural network trained to carry out a particular task. The well known embeddings are those by Mikolov et al. (2013), where the task is to predict the word from its neighboring words (CBOW) or the neighbors from the given word (Skip-gram). Trained on a huge corpus, these "Word2Vec" embeddings show an interesting correspondence between lexical relations and arithmetic operations in the vector space. The most famous example is the following:

$$v(\textit{king}) - v(\textit{man}) + v(\textit{woman}) \approx v(\textit{queen})$$

In other words, adding the vectors associated with the words '*king*' and '*woman*' while subtracting '*man*' should be equal to the vector associated with the word '*queen*'. We can also say that the difference vectors $v(\textit{king}) - v(\textit{queen})$ and $v(\textit{man}) - v(\textit{woman})$ are almost identical and describe the gender relationship.

Word2Vec is not trained with a goal of proper representation of relationships, therefore the absolute accuracy scores around 50% do not allow to rely on these relation predictions. Still, it is a rather interesting property observed empirically in the learned space. Another extensive study of embedding space has been conducted by Hill et al. (2017).

Word2Vec embeddings as well as GloVe embeddings (Pennington et al., 2014) became very popular and they were tested in many tasks, also because for English they can be simply downloaded as pretrained on huge corpora. Word2Vec was trained on 100 billion words Google News

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 56–64

dataset[1] and GloVe embeddings were trained on 6 billion words from the Wikipedia. Sometimes, they are used as a fixed mapping for a better robustness of the system (Kenter and De Rijke, 2015), but they are more often used to seed the embeddings in a system and they are further trained in the particular end-to-end application (Collobert et al., 2011; Lample et al., 2016).

In practice, random initialization of embeddings is still more common than using pretrained embeddings and it should be noted that pretrained embeddings are not always better than random initialization (Dhingra et al., 2017).

We are not aware of any study of the effects of various random embeddings initializations on the training performance.

In the first part of the paper, we explore various English word embeddings initializations in four tasks: neural machine translation (denoted MT in the following for short), language modeling (LM), part-of-speech tagging (TAG) and lemmatization (LEM), covering both common styles of neural architectures: the recurrent and convolutional neural networks, RNN and CNN, resp.

In the second part, we explore the obtained embeddings spaces in an attempt to better understand the networks have learned about word relations.

2 Embeddings initialization

Given a vocabulary V of words, *embeddings* represent each word as a dense vector of size d (as opposed to "one-hot" representation where each word would be represented as a sparse vector of size $|V|$ with all zeros except for one element indicating the given word). Formally, embeddings are stored in a matrix $E \in \mathbb{R}^{|V| \times d}$.

For a given word type $w \in V$, a row is selected from E. Thus, E is often referred to as word lookup table. The size of embeddings d is often set between 100 and 1000 (Bahdanau et al., 2014; Vaswani et al., 2017; Gehring et al., 2017).

2.1 Initialization methods

Many different methods can be used to initialize the values in E at the beginning of neural network training. We distinguish between randomly initialized and pretrained embeddings, where the latter can be further divided into embeddings pretrained on the same task and pretrained on a standard task such as Word2Vec or GloVe.

Random initialization methods common in the literature[2] sample values either uniformly from a fixed interval centered at zero or, more often, from a zero-mean normal distribution with the standard deviation varying from 0.001 to 10.

The parameters of the distribution can be set empirically or calculated based on some assumptions about the training of the network. The second approach has been done for various hidden layer initializations (i.e. not the embedding layer). E.g. Glorot and Bengio (2010) and He et al. (2015) argue that sustaining variance of values thorough the whole network leads to the best results and define the parameters for initialization so that the layer weights W have the same variance of output as is the variance of the input.

Glorot and Bengio (2010) define the "Xavier" initialization method. They suppose a linear neural network for which they derive weights initialization as

$$W \sim \mathcal{U}\big[-\frac{\sqrt{6}}{\sqrt{n_i + n_o}}; \frac{\sqrt{6}}{\sqrt{n_i + n_o}}\big] \qquad (1)$$

where n_i is the size of the input and n_o is the size of the output. The initialization for nonlinear networks using ReLu units has been derived similarly by He et al. (2015) as

$$W \sim \mathcal{N}(0, \frac{2}{n_i}) \qquad (2)$$

The same assumption of sustaining variance cannot be applied to embeddings because there is no input signal whose variance should be sustained to the output. We nevertheless try these initialization as well, denoting them *Xavier* and *He*, respectively.

2.2 Pretrained embeddings

Pretrained embeddings, as opposed to random initialization, could work better, because they already contain some information about word relations.

To obtain pretrained embeddings, we can train a randomly initialized model from the normal distribution with a standard deviation of 0.01, extract embeddings from the final model and use them as pretrained embeddings for the following trainings

[1] See `https://code.google.com/archive/p/word2vec/`.

[2] Aside from related NN task papers such as Bahdanau et al. (2014) or Gehring et al. (2017), we also checked several popular neural network frameworks (TensorFlow, Theano, Torch, ...) to collect various initialization parameters.

on the same task. Such embeddings contain information useful for the task in question and we refer to them as *self-pretrain*.

A more common approach is to download some ready-made "generic" embeddings such as Word2Vec and GloVe, whose are not directly related to the final task but show to contain many morpho-syntactic relations between words (Mikolov et al., 2013; Kocmi and Bojar, 2016). Those embeddings are trained on billions of monolingual examples and can be easily reused in most existing neural architectures.

3 Experimental setup

This section describes the neural models we use for our four tasks and the training and testing datasets.

3.1 Models description

For all our four tasks (MT, LM, TAG, and LEM), we use Neural Monkey (Helcl and Libovický, 2017), an open-source neural machine translation and general sequence-to-sequence learning system built using the TensorFlow machine learning library.

All models use the same vocabulary of 50000 most frequent words from the training corpus. And the size of embedding is set to 300, to match the dimensionality of the available pretrained Word2Vec and GloVe embeddings.

All tasks are trained using the Adam (Kingma and Ba, 2014) optimization algorithm.

We are using 4GB machine translation setup (MT) as described in Bojar et al. (2017) with increased encoder and decoder RNN sizes. The setup is the encoder-decoder architecture with attention mechanism as proposed by Bahdanau et al. (2014). We use encoder RNN with 500 GRU cells for each direction (forward and backward), decoder RNN with 450 conditional GRU cells, maximal length of 50 words and no dropout. We evaluate the performance using BLEU (Papineni et al., 2002). Because our aim is not to surpass the state-of-the-art MT performance, we omit common extensions like beam search or ensembling. Pretrained embeddings also prevent us from using subword units (Sennrich et al., 2016) or a larger embedding size, as customary in NMT. We experiment only with English-to-Czech MT and when using pretrained embeddings we modify only the source-side (encoder) embeddings, because there are no pretrained embeddings available for Czech.

The goal of the language model (LM) is to predict the next word based on the history of previous words. Language modeling can be thus seen as (neural) machine translation without the encoder part: no source sentence is given to translate and we only predict words conditioned on the previous word and the state computed from predicted words. Therefore the parameters of the neural network are the same as for the MT decoder. The only difference is that we use dropout with keep probability of 0.7 (Srivastava et al., 2014). The generated sentence is evaluated as the perplexity against the gold output words (English in our case).

The third task is the POS tagging (TAG). We use our custom network architecture: The model starts with a bidirectional encoder as in MT. For each encoder state, a fully connected linear layer then predicts a tag. The parameters are set to be equal to the encoder in MT, the predicting layer have a size equal to the number of tags. TAG is evaluated by the accuracy of predicting the correct POS tag.

The last task examined in this paper is the lemmatization of words in a given sentence (LEM). For this task we have decided to use the convolutional neural network, which is second most used architecture in neural language processing next to the recurrent neural networks. We use the convolutional encoder as defined by Gehring et al. (2017) and for each state of the encoder, we predict the lemma with a fully connected linear layer. The parameters are identical to the cited work. LEM is evaluated by a accuracy of predicting the correct lemma.

When using pretrained Word2Vec and GloVe embeddings, we face the problem of different vocabularies not compatible with ours. Therefore for words in our vocabulary not covered by the pretrained embeddings, we sample the embeddings from the zero-mean normal distribution with a standard deviation of 0.01.

3.2 Training and testing datasets

We use CzEng 1.6 (Bojar et al., 2016), a parallel Czech-English corpus containing over 62.5 million sentence pairs. This dataset already contains automatic word lemmas and POS tags.[3]

[3]We are aware that training and evaluating a POS tagger and lemmatizer on automatically annotated data is a little questionable because the data may exhibit artificial regularities and cannot lead to the best performance, but we assume

Initialization	MT en-cs (25M)	LM (25M)	RNN TAG (3M)	CNN LEM (3M)
$\mathcal{N}(0, 10)$	6.93 BLEU	76.95	85.2 %	48.4 %
$\mathcal{N}(0, 1)$	9.81 BLEU	61.36	87.9 %	94.4 %
$\mathcal{N}(0, 0.1)$	11.77 BLEU	56.61	90.7 %	95.7 %
$\mathcal{N}(0, 0.01)$	11.77 BLEU	56.37	**90.8 %**	**95.9 %**
$\mathcal{N}(0, 0.001)$	**11.88 BLEU**	**55.66**	90.5 %	**95.9 %**
Zeros	11.65 BLEU	56.34	90.7 %	**95.9 %**
Ones	10.63 BLEU	62.04	90.2 %	95.7 %
He init.	11.74 BLEU	56.40	90.7 %	95.7 %
Xavier init.	11.67 BLEU	55.95	**90.8 %**	**95.9 %**
Word2Vec	12.37 BLEU	**54.43**	90.9 %	95.7 %
Word2Vec on trainset	11.74 BLEU	54.63	90.8 %	95.6 %
GloVe	11.90 BLEU	55.56	90.6 %	95.5 %
Self pretrain	**12.61 BLEU**	54.56	**91.1 %**	**95.9 %**

Table 1: Task performance with various embedding initializations. Except for LM, higher is better. The best results for random (upper part) and pretrained (lower part) embedding initializations are in bold.

We use the `newstest2016` dataset from WMT 2016[4] as the testset for MT, LM and LEM. The size of the testset is 2999 sentence pairs containing 57 thousands Czech and 67 thousands English running words.

For TAG, we use manually annotated English tags from PCEDT[5] (Hajič et al., 2012). From this dataset, we drop all sentences containing the tag "-NONE-" which is not part of the standard tags. This leads to the testset of 13060 sentences of 228k running words.

4 Experiments

In this section, we experimentally evaluate embedding initialization methods across four different tasks and two architectures: the recurrent and convolutional neural networks.

The experiments are performed on the NVidia GeForce 1080 graphic card. Note that each run of MT takes a week of training, LM takes a day and a half and TAG and LEM need several hours each. We run the training for one epoch and evaluate the performance regularly throughout the training on the described test set. For MT and LM, the epoch amounts to 25M sentence pairs and for TAG and LEM to 3M sentences. The epoch size is set empirically so that the models already reach a stable level of performance and further improvement does not increase the performance too much.

MT and LM exhibit performance fluctuation throughout the training. Therefore, we average the results over five consecutive evaluation scores spread across 500k training examples to avoid local fluctuations. This can be seen as a simple smoothing method.[6]

4.1 Final performance

In this section, we compare various initialization methods based on the final performance reached in the individual tasks. Intuitively, one would expect the best performance with self-pretrained embeddings, followed by Word2Vec and GloVe. The random embeddings should perform worse.

Table 1 shows the influence of the embedding initialization on various tasks and architectures.

The rows $ones$ and $zeros$ specify the initialization with a single fixed value.

The "Word2Vec on trainset" are pretrained embeddings which we created by running Gensim (Řehůřek and Sojka, 2010) on our training set. This setup serves as a baseline for the embeddings pretrained on huge monolingual corpora and we can notice a small loss in performance compared to both Word2Vec and GloVe.

We can notice several interesting results. As expected, the self-pretrained embeddings slightly outperform pretrained Word2Vec and GloVe, which are generally slightly better than random initializations.

A more important finding is that there is generally no significant difference in the performance between different random initialization methods, except $ones$ and setups with the standard deviation of 1 and higher, all of which perform considerably worse.

that this difference will have no effect on the comparison of embeddings initializations and we prefer to use the same training dataset for all our tasks.

[4] http://www.statmt.org/wmt16/translation-task.html

[5] https://ufal.mff.cuni.cz/pcedt2.0/en/index.html

[6] See, e.g. `http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc42.htm` from Natrella (2010) justifying the use of the simple average, provided that the series has leveled off, which holds in our case.

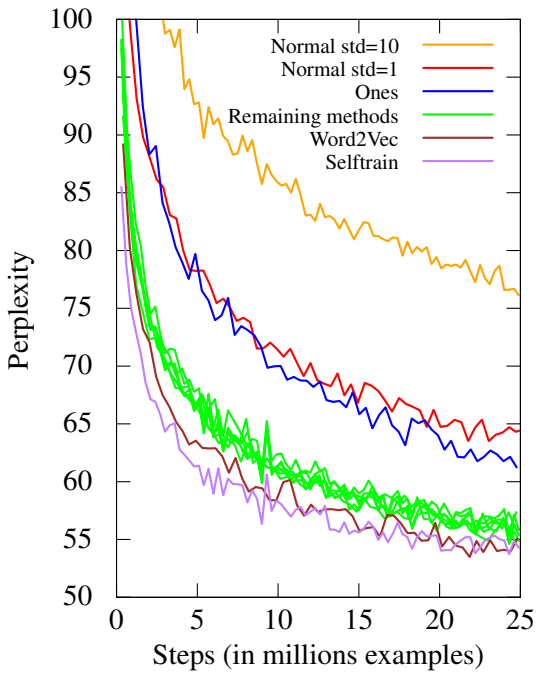


Figure 1: Learning curves for language modeling. The testing perplexity is computed every 300k training examples. Label "Remaining methods" represents all learning curves for the methods from Table 1 not mentioned otherwise.

Any random initialization with standard deviation smaller than 0.1 leads to similar results, including even the $zero$ initialization.[7] We attempt to explain this behavior in Section 5.

4.2 Learning speed

While we saw in Table 1 that most of the initialization methods lead to a similar performance, the course of the learning is slightly more varied. In other words, different initializations need different numbers of training steps to arrive at a particular performance. This is illustrated in Figure 1 for LM.

To describe the situation concisely across the tasks, we set a minimal score for each task and we measure how many examples did the training need to reach the score. We set the scores as follows: MT needs to reach 10 BLEU points, LM needs to reach the perplexity of 60, TAG needs to reach the accuracy of 90% and LEM needs to reach the accuracy of 94%.

We use a smoothing window as implemented in TensorBoard with a weight of 0.6 to smooth the testing results throughout the learning. This way, we avoid small fluctuations in training and our estimate when the desired value was reached is more reliable.

[7] It could be seen as a surprise, that zero initialization works at all. But since embeddings behave as weights for bias values, they learn quickly from the random weights available throughout the network.

Initialization	MT en-cs	LM	TAG	LEM
$\mathcal{N}(0, 1)$	25.3M	37.3M	10.6M	2.7M
$\mathcal{N}(0, 0.1)$	9.7M	13.5M	2.0M	1.8M
$\mathcal{N}(0, 0.01)$	9.8M	12.0M	1.4M	1.2M
$\mathcal{N}(0, 0.001)$	9.8M	12.0M	1.0M	0.5M
Zeros	9.4M	12.3M	1.0M	**0.5M**
Ones	18.9M	26.7M	2.9M	0.8M
He init.	9.5M	12.5M	1.0M	**0.5M**
Xavier init.	9.2M	12.3M	1.0M	**0.5M**
Word2Vec	6.9M	7.9M	0.7M	1.2M
GloVe	8.6M	11.4M	1.9M	1.3M
Self pretrain	**5.2M**	**5.7M**	**0.2M**	0.9M

Table 2: The number of training examples needed to reach a desired score.

The results are in Table 2. We can notice that pretrained embeddings converge faster than the randomly initialized ones on recurrent architecture (MT, LM and TAG) but not on the convolutional architecture (LEM).

Self-pretrained embeddings are generally much better. Word2Vec also performs very well but GloVe embeddings are worse than random initializations for TAG.

5 Exploration of embeddings space

We saw above that pretrained embeddings are slightly better than random initialization. We also saw that the differences in performance are not significant when initialized randomly with small values.

In this section, we analyze how specific lexical relations between words are represented in the learned embeddings space. Moreover, based on the observations from the previous section, we propose a hypothesis about the failure of initialization with big numbers ($ones$ or high-variance random initialization) and try to justify it.

The hypothesis is as follows:

The more variance the randomly initialized embeddings have, the more effort must the neural network exerts to store information in the embeddings space. Above a certain effort threshold, it becomes easier to store the information in the subsequent hidden layers (at the expense of some capacity loss) and use the random embeddings more or less as a strange "multi-hot" indexing mechanism. And on the other hand, initialization with a small variance or even all zeros leaves the neu-

ral network free choice over the utilization of the embedding space.

We support our hypothesis as follows.

- We examine the embedding space on the performance in lexical relations between words, If our hypothesis is plausible, low-variance embeddings will perform better at representing these relations.

- We run an experiment with non-trainable fixed random initialization to demonstrate the ability of the neural network to overcome broken embeddings and to learn the information about words in its deeper hidden layers.

5.1 Lexical relations

Recent work on word embeddings (Vylomova et al., 2016; Mikolov et al., 2013) has shown that simple vector operations over the embeddings are surprisingly effective at capturing various semantic and morphosyntactic relations, despite lacking explicit supervision in these respects.

The testset by Mikolov et al. (2013) contains "questions" defined as $v(X) - v(Y) + v(A) \sim v(B)$. The well-known example involves predicting a vector for word '*queen*' from the vector combination of $v(\textit{king}) - v(\textit{man}) + v(\textit{woman})$. This example is a part of "semantic relations" in the test set, called opposite-gender. The dataset contain another 4 semantic relations and 9 morphosyntactic relations such as pluralisation $v(\textit{cars}) - v(\textit{car}) + v(\textit{apple}) \sim v(\textit{apples})$.

Kocmi and Bojar (2016) revealed the sparsity of the testset and presented extended testset. Both testsets are compatible and we use them in combination.

Note that the performance on this test set is affected by the vocabulary overlap between the test set and the vocabulary of the embeddings; questions containing out-of-vocabulary words cannot be evaluated. This is the main reason, why we trained all tasks on the same training set and with the same vocabulary, so that their performance in lexical relations can be directly compared.

Another lexical relation benchmark is the word similarity. The idea is that similar words such as '*football*' and '*soccer*' should have vectors close together. There exist many datasets dealing with word similarity. Faruqui and Dyer (2014) have extracted words similarity pairs from 12 different corpora and created an interface for testing the embeddings on the word similarity task.[8]

Initialization	MT en-cs	LM	LEM
$\mathcal{N}(0, 10)$	0.0; 0.3	0.0; 0.3	0.0; 0.3
$\mathcal{N}(0, 1)$	0.0; 0.4	1.4; 3.5	0.0; 0.3
$\mathcal{N}(0, 0.1)$	1.2; 23.5	5.5; 15.2	0.0; 0.8
$\mathcal{N}(0, 0.01)$	2.0; 29.9	6.9; 19.4	0.1; 32.7
$\mathcal{N}(0, 0.001)$	2.1; 31.4	6.7; 18.2	0.3; 33.3
Zeros	1.6; 29.5	6.0; 17.5	0.2; 31.1
Ones	0.5; 16.6	5.3; 9.3	0.1; 31.0
He init.	1.4; 28.9	7.7; 18.3	0.1; 32.6
Xavier init.	1.5; 29.5	7.4; 18.2	0.1; 32.7
Word2Vec on trainset*	22.3; 48.9		
Word2Vec official*	81.3; 70.7		
GloVe official*	12.3; 60.1		

Table 3: The accuracy in percent on the (semantic; morphosyntactic) questions. We do not report TAG since its accuracy was less than 1% on all questions. *For comparison, we present results of Word2Vec trained on our training set and official trained embeddings before applying them on training of particular task.

Initialization	MT en-cs	LM	TAG	LEM
$\mathcal{N}(0, 10)$	3.3	2.2	3.6	2.6
$\mathcal{N}(0, 1)$	15.7	11.8	3.5	2.7
$\mathcal{N}(0, 0.1)$	56.7	32.7	6.9	2.8
$\mathcal{N}(0, 0.01)$	62.5	41.0	12.8	4.7
$\mathcal{N}(0, 0.001)$	59.3	37.4	12.1	2.4
Zeros	57.9	37.4	12.8	3.5
Ones	34.0	19.3	11.4	4.3
He init.	58.2	37.4	12.3	4.2
Xavier init.	58.3	37.5	12.3	2.7

Table 4: Spearman's correlation ρ on word similarities. The results are multiplied by 100.

When evaluating the task, we calculate the similarity between a given pair of words by the cosine similarity between their corresponding vector representation. We then report Spearmans rank correlation coefficient between the rankings produced by the embeddings against human rankings. For convenience, we combine absolute values of Spearman's correlations from all 12 Faruqui and Dyer (2014) testsets together as an average weighted by the number of words in the datasets.

The last type of relation we examine are the nearest neighbors. We illustrate on the TAG task how the embedding space is clustered when various initializations are used. We employ the Principal component analysis (PCA) to convert the embedding space of $|E|$ dimensions into two-dimensional space.

Table 3 reflects several interesting properties

[8] http://wordvectors.org/

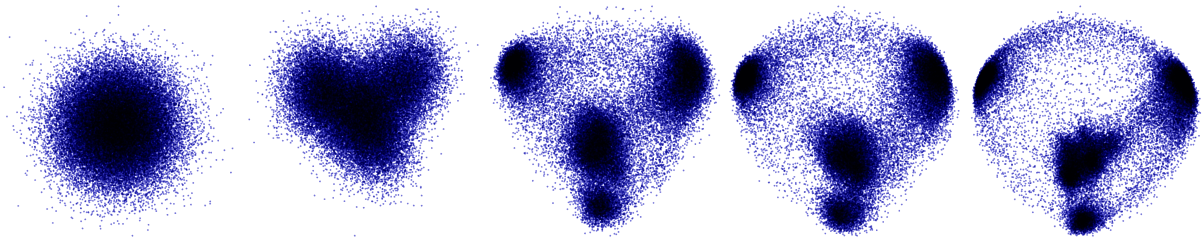


Figure 2: A representation of words in the trained embeddings for TAG task projected by PCA. From left to right it shows trained embeddings for $\mathcal{N}(0, 1)$, $\mathcal{N}(0, 0.1)$, $\mathcal{N}(0, 0.01)$, $\mathcal{N}(0, 0.001)$ and $zeros$. Note that except of the first model all of them reached a similar performance on the TAG task.

about the embedding space. We see task-specific behavior, e.g. TAG not learning any of the tested relationships whatsoever or LM being the only task that learned at least something of semantic relations.

The most interesting property is that when increasing the variance of initial embedding, the performance dramatically drops after some point. LEM reveals this behavior the most: the network initialized by normal distribution with standard deviation of 0.1 does not learn any relations but still performs comparably with other initialization methods as presented in Table 1. We ran the lemmatization experiments once again in order to confirm that it is not only a training fluctuation.

This behavior suggests that the neural network can work around broken embeddings and learn important features within other hidden layers instead of embeddings.

A similar behavior can be traced also in the word similarity evaluation in Table 4, where models are able to learn to solve their tasks and still not learn any information about word similarities in the embeddings.

Finally, when comparing the embedded space of embeddings as trained by TAG in Figure 2, we see a similar behavior. With lower variance in embeddings initialization, the learned embeddings are more clearly separated.

This suggests that when the neural network has enough freedom over the embeddings space, it uses it to store information about the relations between words.

5.2 Non-trainable embeddings

To conclude our hypothesis, we demonstrate the flexibility of a neural network to learn despite a broken embedding layer.

In this experiment, the embeddings are fixed and the neural network cannot modify them during the training process. Therefore, it needs to find a way to learn the representation of words in other hidden layers.

Initialization	MT en-cs	LM	TAG	LEM
$\mathcal{N}(0, 10)$	7.28 BLEU	79.44	47.3 %	85.5 %
$\mathcal{N}(0, 1)$	8.46 BLEU	78.68	87.1 %	94.0 %
$\mathcal{N}(0, 0.01)$	6.84 BLEU	82.84	63.2 %	91.1 %
Word2Vec	8.71 BLEU	60.23	88.4 %	94.1 %

Table 5: The results of the experiment when learned with non-trainable embeddings.

As in Section 4.1, we train models for 3M examples for TAG and LEM and for over 25M examples for MT and LM.

Table 5 confirms that the neural network is flexible enough to partly overcome the problem with fixed embeddings. For example, MT initialized with $\mathcal{N}(0, 1)$ reaches the score of 8.46 BLEU with fixed embeddings compared to 9.81 BLEU for the same but not fixed (trainable) embeddings.

When embeddings are fixed at random values, the effect is very similar to embeddings with high-variance random initialization. The network can distinguish the words through the crippled embeddings but has no way to improve them. It thus proceeds to learn in a similar fashion as with one-hot representation.

6 Conclusion

In this paper, we compared several initialization methods of embeddings on four different tasks, namely: machine translation (RNN), language modeling (RNN), POS tagging (RNN) and lemmatization (CNN).

The experiments indicate that pretrained embeddings converge faster than random initialization and that they reach a slightly better final performance.

The examined random initialization methods do not lead to significant differences in the performance as long as the initialization is within reasonable variance (i.e. standard deviation smaller than 0.1). Higher variance apparently prevents the network to adapt the embeddings to its needs and the network resorts to learning in its other free parameters. We support this explanation by showing that the network is flexible enough to overcome even non-trainable embeddings.

We also showed a somewhat unintuitive result that when the neural network is presented with embeddings with a small variance or even all-zeros embeddings, it utilizes the space and learns (to some extent) relations between words in a way similar to Word2Vec learning.

Acknowledgement

This work has been in part supported by the European Union's Horizon 2020 research and innovation programme under grant agreements No 644402 (HimL) and 645452 (QT21), by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (projects LM2015071 and OP VVV VI CZ.02.1.01/0.0/0.0/16_013/0001781), by the Charles University Research Programme "Progres" Q18+Q48, by the Charles University SVV project number 260 453 and by the grant GAUK 8502/2016.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. 2016. Czeng 1.6: Enlarged czech-english parallel corpus with processing tools dockered. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech, and Dialogue: 19th International Conference, TSD 2016*, number 9924 in Lecture Notes in Computer Science, pages 231–238. Masaryk University, Springer International Publishing.

Ondřej Bojar, Jindřich Helcl, Tom Kocmi, Jindřich Libovický, and Tomáš Musil. 2017. Results of the WMT17 Neural MT Training Task. In *Proceedings of the 2nd Conference on Machine Translation (WMT)*, Copenhagen, Denmark, September.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Bhuwan Dhingra, Hanxiao Liu, Ruslan Salakhutdinov, and William W. Cohen. 2017. A comparative study of word embeddings for reading comprehension. *CoRR*, abs/1703.00993.

Manaal Faruqui and Chris Dyer. 2014. Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of ACL: System Demonstrations*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the Eighth International Language Resources and Evaluation Conference (LREC'12)*, pages 3153–3160, Istanbul, Turkey, May. ELRA, European Language Resources Association.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Jindřich Helcl and Jindřich Libovický. 2017. Neural monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics*, 107:5–17.

Felix Hill, Kyunghyun Cho, Sébastien Jean, and Yoshua Bengio. 2017. The representational geometry of word meanings acquired by neural machine translation models. *Machine Translation*, pages 1–16.

Tom Kenter and Maarten De Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1411–1420. ACM.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Tom Kocmi and Ondřej Bojar, 2016. *SubGram: Extending Skip-Gram Word Representation with Substrings*, pages 182–189. Springer International Publishing.

Guillaume Lample, Miguel Ballesteros, Kazuya Kawakami, Sandeep Subramanian, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *In proceedings of NAACL-HLT (NAACL 2016)*., San Diego, US.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *CoRR*, abs/1610.03017.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *NAACL Workshop on Vector Space Modeling for NLP*, Denver, United States.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mary Natrella. 2010. Nist/sematech e-handbook of statistical methods.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL 2002, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention is all you need. *ArXiv e-prints*, jun.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1671–1682, Berlin, Germany, August. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Curriculum Design for Code-switching: Experiments with Language Identification and Language Modeling with Deep Neural Networks

Ashutosh Baheti, Sunayana Sitaram, Monojit Choudhury, Kalika Bali
Microsoft Research Lab, India
ashutosh.baheti95@gmail.com
{t-susita, monojitc, kalikab}@microsoft.com

Abstract

Curriculum learning strategies are known to improve the accuracy, robustness and convergence rate for various language learning tasks using deep architectures (Bengio et al., 2009). In this work, we design and experiment with several training curricula for two tasks – word-level language detection and language modeling – for *code-switched* text data. Our study shows that irrespective of the task or the underlying DNN architecture, the best curriculum for training the code-switched models is to first train a network with monolingual training instances, where each mini-batch has instances from both languages, and then train the resulting network on code-switched data.

1 Introduction

Code-switching (CS) refers to the linguistic phenomenon of fluid alternation between two or more languages during a single conversation or even an utterance (Myers-Scotton, 1993). It is observed in all stable multilingual societies (Auer, 1995) and recent studies have shown that social media posts from such societies almost always contain small to moderate amount of CS (Bali et al., 2014; Dorleijn, 2016; Molina et al., 2016; Rudra et al., 2016; Rijhwani et al., 2017). For instance, Rijhwani et al. (2017) shows that 2-12% (3.5% on average) of the tweets from the cities around the world are code-switched. It is therefore imperative to build speech and text processing technologies that can handle CS. Indeed, quite some amount of effort is being invested towards technology for CS (see Diab et al. (2014; 2016), Sharma et al. (2015), and references therein).

It is of theoretical and practical interest to ponder on the question: whether for a particular NLP task (say ASR, MT or POS Tagging), it is possible to build CS models only from pretrained monolingual models or monolingual training data? Indeed, several studies in the past (Solorio and Liu, 2008; Vyas et al., 2014; Gadre et al., 2016; Gonzalez-Dominguez et al., 2015) have proposed techniques for combining monolingual models or training data coupled with a little amount of CS data to build models of CS text or speech. These techniques have reported promising results. However, all these studies, except (Johnson et al., 2016; Rijhwani et al., 2017; Chan et al., 2009), have tried to combine the outputs of pre-trained monolingual models in intelligent ways. On the other hand, one might ask whether a single system trained on monolingual data from both the languages would be able to handle CS between these languages? And, if we also had a little amount of CS data, how best to use it during the training process?

In this paper, we explore various training strategies, also known as *Curriculum* (Bengio et al., 2009) for DNN-based architectures for code-switching. In particular, we design a set of strategies or curricula involving various ordering of the monolingual and CS data. We experiment with these curricula for *Language Identification* (LID) and *Language Modeling* (LM) tasks. Our study shows that the best curriculum across the tasks as well as DNN architecture is the same one: first train a network with monolingual instances alternating between the languages, and then train the resultant network with CS data, if available. The models trained solely with monolingual data also achieve reasonably high accuracies.

As far as we know, this is the first study on curriculum design for CS. Our study has two important implications: first, it shows that it is possible to train models for CS using primarily monolin-

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 65–74

gual data; this obviates the need for creation of large amount of CS datasets. Second, it also brings out the fact that training curriculum is extremely important while building CS models from monolingual data, and there seems to be an ideal way of ordering the training examples that works best across tasks and network structures.

2 Background and Motivation

In this section, we present a typology of the monolingual model combination strategies for CS, through which we will motivate the central idea of this work.

2.1 A Note on Terms

It is important to differentiate between inter-sentential and intra-sentential CS. The former refers to a situation where each sentence (or sometimes clause) is in a single language, but the language might change across the sentences. On the other hand, intra-sentential CS, which is also sometimes called *Code-mixing*, refers to a situation where words in the same sentence/clause can be drawn from multiple languages.

Tasks that operate on sentence level context (like POS tagging, ASR and MT) do not require any special technique for handling inter-sentential CS, except LID and sentence boundary detection. However, intra-sentential CS is more challenging to handle, and will be our primary focus. In this paper, the terms *monolingual model* and *monolingual data* will be used for cases where the data was collected and the model was built assuming that the input will be only in a single language. Such datasets might also contain some borrowed words and text in other language(s). On the other hand, we will use the term *CS data* to imply datasets where all instances contain intra-sentential CS, even though most of the datasets released in the past for training CS models, e.g., (Molina et al., 2016; Das, 2016; Sequiera et al., 2015b), do contain fair amounts of monolingual and inter-sentential CS. The term *CS model* will be used for systems that can handle monolingual, inter-sentential as well as intra-sentential CS.

2.2 A Taxonomy of CS Models

In order to succinctly represent the various types of CS models proposed in the literature, we will use the following notation. Let l_1 and l_2 be two languages. Let x denote the input string, usually a sentence, i.e., string of tokens, in l_1, l_2 or l_{12}, i.e., $l_1 \leftrightarrow l_2$ code-switched. Let y be the output string of tokens in a target language (as in MT, ASR or POS tagging). Let g_i and f_i denote models trained on data from l_i. Further, we describe a special function $lid(x)$ which returns the string of language labels for each word; $lid_1(x)$ and $lid_2(x)$ are projection functions which returns only those tokens of x that are in l_1 and l_2 respectively.

CS models described in the literature can be broadly categorized into the following four classes (in descending order of amount of CS data required for training).

Purely Supervised Models: When a large amount of annotated CS data is available, a supervised model can be learnt simply from the monolingual and CS data. Thus,

$$y = g_{1\cup2\cup12}(x) \tag{1}$$

These models often use features or extra information specific to CS, but do not particularly modify the training process or system architecture for handling CS. This approach has been applied to language identification, e.g., most submissions in the LID shared task in the Computational Approaches to Code-Switching Workshops (Solorio et al., 2014; Molina et al., 2016); to POS tagging, e.g., most submissions in the ICON 2016 shared task on CS POS tagging (Das, 2016) and also (Jamatia and Das, 2014; Jamatia et al., 2015); and to ASR (Gebhardt, 2011).

Combining Monolingual Models: In this approach, the output of two monolingual systems on x is used as features for a third model (f_{12} in Eq. 2). This third model f is trained on a small amount of CS data, and can use other features which often includes LID output.

$$y = f_{12}(g_1(x), g_2(x), lid(x)) \tag{2}$$

Solorio and Liu (2008) proposed this architecture for POS tagging of English-Spanish CS data, and Lyu et al. (2006) proposed a similar model for ASR. Both reported significant gain over the monolingual models by using very little CS data. Later works, such as (Sequiera et al., 2015a), along this line also reported promising results.

Divide and Conquer: In this approach, the input is first passed through a LID system and split into parts according to the language of the tokens. The token strings are then passed on to the respective monolingual systems and the outputs are combined (shown as the operator $\oplus$ in Eq. 3)

$$y = g_1(lid_1(x)) \oplus g_2(x)(lid_2(x)) \quad (3)$$

This approach does not require any CS training data, but it does not work well for intra-sentential CS because splitting by language can lead to loss of context especially at the code-switch points. However, some benefits of this approach have been shown for POS tagging (Vyas et al., 2014), MT (Gadre et al., 2016) and ASR (Lyudovyk and Pylypenko, 2014) respectively.

Zero Shot Learning: This is an extreme case, where only monolingual data from two or more languages is used to train a single system with the hope that it will work for CS data as well.

$$y = g_{1\cup 2}(x) \quad (4)$$

A recent work (Rijhwani et al., 2017) uses this technique very effectively for developing an LID system for 7 languages. While no annotated CS data is used for training, the system uses unlabeled data that is expected to contain CS data, for unsupervised training. Johnson et al. (2016) trains a neural MT system with data from two pairs of languages, $l_1 \Leftrightarrow l_2$ and $l_1 \Leftrightarrow l_3$ and show that the resultant model not only works for $l_2 \Leftrightarrow l_3$ (the so called "zero shot learning" but also for CS input in these languages, albeit to a limited extent.

Factors such as lack of large-scale CS datasets, possibility of CS between any pair (or even triplet) of languages (which in turn implies the need for nearly a quadratic number of such datasets) and the difficulty in creation of CS datasets owing to the requirement of skilled multilingual annotators make Zero Shot Learning a very lucrative approach CS. However, we do not know of any work that systematically explores the various training strategies and effective use of CS data in the context of Zero Shot Learning.

3 Training Curricula for CS

Originally proposed by (Elman, 1993), *Curriculum learning* refers to a sequence of weight distributions over the training example, such that during the training process certain examples are used with higher weight at the initial stages of the training and other examples are used later (Bengio et al., 2009). In general, it is believed that for complex non-convex optimization problems, training with *simpler* examples first and introducing the complex examples at later stage has distinctive benefits. Empirically, it has shown promising results for several NLP tasks like parsing (Spitkovsky et al., 2009) and language modeling (Bengio et al., 2009; Graves et al., 2017). Shi et al. (2015) describe curricula for domain adaptation of Language Models, in which they order the data such that general data is presented to the RNN first, followed by in-domain data.

In principle, the purely supervised (Eq. 1) and the zero shot learning approaches (Eq. 4) should benefit from curriculum based training. It is well known that proficient bilingual children learn to code-switch without any exposure to CS data (Cantone, 2007). This leads us to explore various curricula for training with monolingual and CS data. While complexity of training instances can be defined across various dimensions, in this study we will restrict ourselves to only one aspect of the curriculum design - the language(s).

Let T_1, T_2 and T_{12} be respectively the set of training examples in l_1, l_2 and intra-sentential CS between l_1 and l_2. We will use the notation $T_i; T_j$ to indicate a basic curriculum where the system is trained with all instances from T_i first, and then with instances from T_j. Similarly, $\{T_i, T_j\}$ will be used to indicate the curiculum where the system is trained with instances from T_i and T_j simultaneously; in the context of deep learning, this means each *mini-batch* contains samples from T_i and T_j (ideally, but not necessarily, in a ratio $|T_i| : |T_j|$).

Based on the ordering of the training instances, we define 7 different curricula: (C1) $T_1; T_2$ (C2) $T_{12}; T_1; T_2$ (C3) $T_1; T_2; T_{12}$ (C4) $\{T_1, T_2\}$ (C5) $T_{12}; \{T_1, T_2\}$ (C6) $\{T_1, T_2\}; T_{12}$ (C7) $\{T_1, T_2, T_{12}\}$. We consider the curricula (C0) T_{12} (i.e., training with only CS data) and C7 (i.e., all instances randomized[1]) as the two baselines.

In the next two sections, we will describe experiments with these curricula for deep learning models applied to two tasks – Language Identification (LID) and Language Modeling (LM), both for English (En) and Spanish (Es) CS.

4 Language Identification

Along the lines of (Rijhwani et al., 2017), we define word-level LID as a sequence labeling problem, where each token in the input sentence is labeled with one of the three tags: l_1, l_2 and X (meaning "none of the languges", such as numbers, punctuations, urls and hashtags). In the

[1]Strictly speaking, in C7 we ensure that in each mini-batch there are training instances from T_1, T_2 and T_{12} in certain fixed ratio.

Language	Train	Dev	Test
En	1240k	1006	13542
Es	1240k	1119	4874
En-Es	17.5k	750	8678

Table 1: Datasets for LID (in number of words).

following subsections, we describe the datasets, DNN architecture, and experimental results.

4.1 Datasets

All our experiments are done on En and Es tweets, which are primarily drawn from two existing datasets: (Rijhwani et al., 2017) for monolingual training data, and (Solorio et al., 2014) for CS training data, and all dev and test datasets. Since we define LID as a classification problem, we consider each word with its context as an instance (instead of each tweet as an instance). Further, we differentiate between CS and monolingual instances as those where the context (a window of $2k$ words around the target word) has or does not have a code-switch point, respectively. Table 1 summarizes the size of the datasets. There are total 1328 switching points in the CS test data.

4.2 DNN Architecture

Fig. 1 shows the architecture of the DNN for LID. The model takes $2k + 1$ word window input with target word at the center, and predicts the language of the target word. In this word-context block, all the input words are projected into a d_w dimensional word-embedding space. The embedding vectors of the k words in the left and right contexts are averaged separately. These left-average, right-average and the current word embedding are merged into one $3d_w$ dimensional vector. This vector is passed to an intermediate Dense layer (with ReLu activation) of d_{im} dimension. To speed up the convergence of the network, we add a batch normalization layer. Also, a dropout layer with 0.3 probability is introduced to prevent overfitting. Finally, a softmax layer with two nodes, one for each language, l_1 and l_2, is used for predicting the output.

We define another architecture - the *Char-LID model* where this basic LID model is augmented with a character-context block (as shown in the dashed box in the Fig. 1). We embed character tri-grams into a d_c dimensional space; an average of all the character tri-grams of the target word is then concatenated with the embeddings of

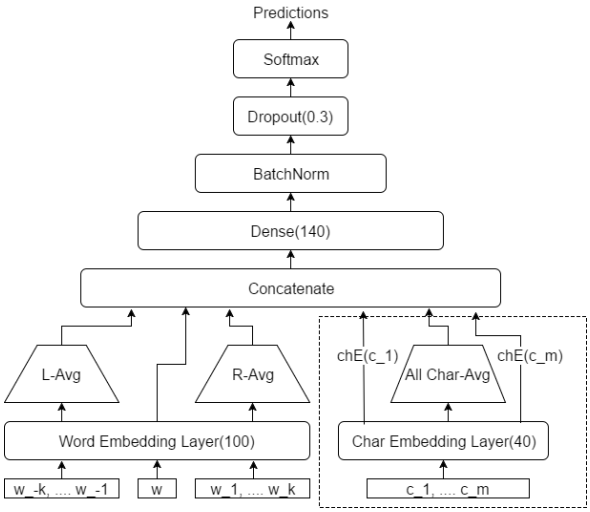


Figure 1: DNN achitecture for LID

the first and last trigrams to generate a $3d_c$ dimensional character-representation vector of the target word. In the Char-LID model, this vector is concatenated to the aforementioned $3d_w$ dimensional word vector, and is fed as the input to the intermediate dense layer. Rest of the network is identical to the LID model. We expect the Char-LID model to work better for out-of-vocabulary words.

4.3 Experiments

The networks were implemented, trained and tested using the Microsoft's Cognitive Toolkit (CNTK)(Yu et al., 2014). For all our experiments, we set[2] $k = 3$, $d_w = 100$, $d_c = 40$, $d_{im} = 140$, and dropout rate = 0.3. The networks are trained using mean square error loss and Adadelta SGD under default parameter settings of CNTK. The word and character vectors were initialized uniformly randomly.

Note that we do not use the DNN to predict the X labels, as these are identified through regular expressions during the pre-processing of the data and are never used as target words during training or testing. Details of how we handle the special and boundary cases (e.g., out-of-vocabulary words and contexts for target words in the beginning and end of a sentence) are discussed in the supplementary material.

For each curriculum, we train the models for 20 epochs and choose the model that has maximum overall accuracy on the dev set. For curricula involving interleaving of instances of different types (e.g., C4 to C7), we presented the training data

[2]Experiments with different values of these parameters led us to these numbers which seem to work well.

Curriculum	All	En-Es	En	Es
LID Model				
C0: T_{12}	89.1	86.6	89.1	93.4
C1: $T_1; T_2$	37.0	48.3	17.8	70.3
C2: $T_{12}; T_1; T_2$	35.4	54.2	0.1	100
C3: $T_1; T_2; T_{12}$	84.1	78.0	95.5	63.6
C4: $\{T_1, T_2\}$	94.7	87.0	99.6	96.3
C5: $T_{12}; \{T_1, T_2\}$	94.8	86.7	99.3	96.7
C6: $\{T_1, T_2\}; T_{12}$	**95.1**	**89.1**	98.8	95.8
C7: $\{T_1, T_2, T_{12}\}$	94.4	87.7	97.5	97.4
Char-LID Model				
C4: $\{T_1, T_2\}$	95.5	87.6	99.7	97.7
C5: $T_{12}; \{T_1, T_2\}$	95.5	87.3	99.7	98.4
C6: $\{T_1, T_2\}; T_{12}$	**97.1**	**93.7**	98.7	98.3
C7: $\{T_1, T_2, T_{12}\}$	96.2	89.6	99.7	98.2

Table 2: Curriculum training accuracies (in %) for LID and Char-LID models. The maximum accuracy for the models are in bold and are statistically significantly higher ($p < 0.001$) than all other values in the column for that model.

to the network in randomized order. Thus, every minibatch is expected to contain the instances in ratio $|T_1| : |T_2|$. Since the $|T_{12}|$ training data is significantly low compared to $|T_1|$ and $|T_2|$, we oversample T_{12} by replicating the data 10 times for curriculum C7. For curricula that involves ordering of inputs by blocks (all except C4 and C7), we first train on the first block of instances for 20 epochs and choose the best model which is trained on the next block of instances for 20 epochs.

4.4 Results

In Table 2 we report the l_1 and l_2 labeling accuracy on the En, Es and En-Es CS test sets, as well as the combined accuracy (column 1) on all the test instances. Due to the significantly poorer performance of C1, C2 and C3 for the LID model, these experiments were not conducted for the char-LID model. For C0, overall accuracy for the Char-LID model is 94.5% (Table 3).

The key observations from Table 2 are: (a) curriculum C6 is most effective across the models; the performance on CS set increases significantly with only a marginal drop in accuracy on the monolingual data; (b) C6 achieves an 11% (55%) and 23% (47%) error reduction over the baseline curriculum C7 (C0) for the LID and Char-LID models respectively; (c) The improvements are highly significant at $p < 0.001$ for a paired t-test, which implies that C6 is able to correct labeling errors made by C7 and other curricula, and hardly makes new labeling errors; (d) Providing the CS training data as the last block is much more effective than providing it in the beginning or distributing it over the entire training curriculum; (e) On the other hand, mixing of monolingual data is more effective than providing them in block; (f) Finally, it is also interesting to note that curriculum C4, where only monolingual training data is used, achieves reasonably high accuracies.

We manually inspected around 100 erroneously labeled words by the best C6 model. There are three noticeable error patterns: (a) errors around an X tag (approx. 15%), (b) errors at or near the interjections such as "haha", "jaja", "lmao" etc. (approx. 15%), and (c) errors near the code-switch point and sentence boundaries (approx. 25%). Rest of the errors didn't have any noticeable pattern, though we also discovered that some of the system labels classified as errors were actually correct and rather the gold standard label was incorrect (approx. 5%).

We also conduct two auxiliary sets of experiments to understand the effect of the ratio of monolingual training instances ($|T_1| : |T_2|$) and that of the CS data to monolingual data ($|T_{12}| : |T_1 \cup T_2|$). In the first experiment, we train the LID model where we vary the percentage of training instances used from the En and Es monolingual datasets. The results are shown in Fig. 2. In the x-axis, we plot the percentage of training instances used from the En and Es training sets during each experiment, where En fractions (the first value in the tuple) increases from right to left, Es fractions (the second value) from left to right. It is evident from the plot that the system performance is not strongly sensitive to the ratio of $|T_1| : |T_2|$. Rather, it is more sensitive to the absolute amount of data available for training; when the data for either T_1 or T_2 drops significantly (less than 1% of the training set here, as in the extremes of the plot), the accuracy is affected significantly.

In the second set of experiments, we train the Char-LID model with curriculum C7. We vary the monolingual and CS training data independently and report the accuracies on the entire test set for each setup in Table 3. The trends, as expected, shows diminishing marginal utility for both monolingual and CS datasets. Nevertheless, we note that the marginal utility of monolingual data is

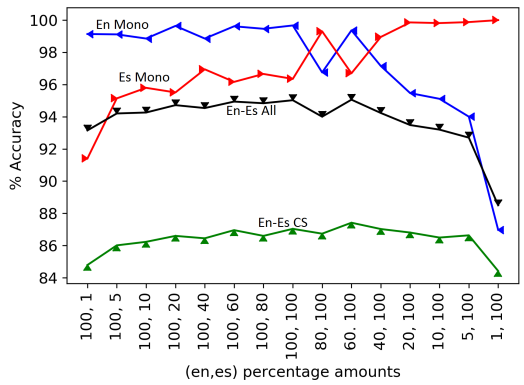


Figure 2: en-es Skew experiment using Curriculum C4 in LID model

% of $\lvert T_{12} \rvert$	% of $\lvert T_1 \rvert$ and $\lvert T_2 \rvert$			
	0	**0.1**	**1**	**10**
20	89.5	93.3	94.4	95.8
50	92.5	93.6	95.4	95.9
100	94.5	93.3	95.1	96.1

Table 3: Overall accuracy of the Char-LID model for C7 with varying amounts of training data.

much more pronounced (i.e., systematic increase of accuracy in each row from left to right) than that of CS data. This could be because of much higher sizes of the monolingual training sets as compared to the CS dataset.

We have also conducted some of these experiments with English-French CS data, which shows similar trends. Reader may refer to the supplementary material for more details.

4.5 Related Work on LID

There have been many works on LID for CS. See (Solorio et al., 2014; Molina et al., 2016; Rijhwani et al., 2017) and references therein. Two systems (Samih et al., 2016; Jaech et al., 2016) submitted in the shared task on language detection in EMNLP 2016 use deep learning based techniques. Samih et al. (2016) uses LSTMs on top of word and character context with a CRF classifier and achieves an accuracy of 96.3% on the same En-Es test set. Jaech et al. (2016) uses only the character sequence data with stacked CNNs to create a word embedding. Then it creates a global context by adding a bi-directional LSTM on top of it. Their system achieves 94.6 average F1 score for En-Es. Chang and Lin (2014 describes an RNN based system which is trained and tested on the EMNLP 2014 shared task dataset. This system outperforms all the submitted systems in that shared task.

The EMNLP shared task dataset had 6 labels including named entities and mixed language words which we did not consider in this work. Therefore, even though we evaluate on the same dataset, the accuracies are not directly comparable. However, since majority of the tokens are labeled En and Es, our results are certainly comparable to these state-of-the-art systems. Rijhwani et al. (2017) uses HMM model initialized by monolingual data and retrained it on unlabeled data using Baum-Welch algorithm. It achieves an average F-1 score of 97.8% for En and Es labels, which is only slightly better than our best performing system. However, the models described here do not use the unlabeled data, incorporating which could be an interesting future direction for this research.

5 Language Modeling

Statistical Language Models estimate the probability of a word sequence given a large training corpus. Language Modeling has applications in various NLP and Speech processing tasks, most notably in MT and ASR. In this section, we describe experiments on building En-Es code-switched LM.

5.1 Datasets

Similar to our LID experiment datasets described in Table 1, we use the En and Es monolingual tweets from (Rijhwani et al., 2017) (described as the *weakly-labeled data* in the paper), and the language-labeled CS data from (Solorio et al., 2014) for training. However, unlike the LID experiments, here a tweet, rather than a word, is considered as an instance; any tweet with CS is a part of the CS training instance. Since we do not need the language labels for the LM training experiments, the tweets were stripped off those labels. We used 212k En tweets, 92k Es tweets and 798 En-Es CS tweets as training data. The amount of CS data used is a very small fraction (0.2%) of the size of the monolingual data and the amount of En data is more than double the Es data.

We also created a held-out test set for the LM experiments, which consists of 2200 En-Es tweets that were automatically tagged as code-switched by our best En-Es LID system described in the previous section. We use the standard evaluation metric – *perplexity* (PPL) on the held-out test set

– to compare the LMs (lower the better).

5.2 Models

We train both RNNLMs and ngram language models on the same data to compare their performances on various training data curricula.

We use the RNNLM toolkit (Mikolov et al., 2011) to train and test all our RNNLM models. The ngram models are also built with the same toolkit, which invokes the SRILM toolkit (Stolcke et al., 2002). During our initial experiments on finding the appropriate curriculum for training CS LMs, we use the single iteration recipe provided in the RNNLM toolkit. In this setting, the learning rate is manually set to decrease after 4 epochs and no validation data is used.

We then use the regular LM training recipe in the RNNLM toolkit that makes use of a validation data set. We adjust the values of various hyperparameters in our experiments. One crucial parameter we adjust is the number of classes (700), according to a rule of thumb saying that the number of classes should be approximately the square root of the vocabulary size. By doing this, we sacrifice accuracy for speed of training our models; since the aim here is to analyze the trends rather than look at the absolute values, we believe this is a reasonable policy. Our models have a very large vocabulary size owing to the presence of two languages and also due to large amount of spelling variations found in tweets. For the experiments with a single iteration, our models have 150 hidden layers. Other hyper parameters are kept at their default values in the corresponding recipes in the RNNLM toolkit. In our final experiments with the full iteration recipe, we used CS data as validation and 100 hidden layers.

5.3 Experiments

We experiment with all the training curricula described in Sec. 3, except C5 because from our experiments with C2 and C3, we find that adding CS data at the beginning produces worse results. For experiments involving $\{T_1, T_2\}$ (i.e., curricula C4 and C6), we provide the input by interleaving the tweets from T_1 and T_2. Since En has almost twice as many tweets as Es, the extra En tweets that remain after interleaving all the Es tweets, are simply appended to the training set. Finally, since we have very little CS data as compared to the monolingual data, for curriculum C7 we replicate the entire T_{12} dataset after every 10k instances from $\{T_1, T_2\}$. Thus, we use 30 replicas of $\{T_{12}\}$ in this curriculum. In addition to these curricula, we also build a baseline using only CS data (C0).

We build all the six models using the RNNLM single iteration setting first. Then, we build models for the best performing curriculum using the multiple iteration recipe. In addition, we build 5-gram models for all the 6 curricula and the C0 baseline. However, since ngram models do not depend on the ordering, there are only four unique corpora for training them: (1) with only monolingual data: $T_1 \cup T_2$, which is comparable to the models for curricula C1 and C4; (2) with monolingual and CS data without replication: $T_1 \cup T_2 \cup T_{12}$, which is comparable to RNNLMs built using curricula C2, C3 and C6; and (3) monolingual data and with replicated CS data[3]: $T_1 \cup T_2 \cup 30 \cdot T_{12}$, which is comparable to the RNNLM trained using curriculum C7 (4) only code-switched data T_{12}, which is comparable to the RNNLM trained using only CS data (comparable to C0).

5.4 Results

Table 4 shows the results of the LM experiments in terms of perplexity on the held-out test set. As we see from the numbers in first column, the best performing RNNLMs are those that are trained initially with monolingual data and then trained with CS data (C3 and C6). The model that is trained initially with CS data and then with monolingual data (C2) performs as badly as the model that was trained with only monolingual data in blocks (C1). In addition, training models with alternate monolingual sentences gives better performance (C4, C5) than training it with large chunks of monolingual text (C1, C2). Training with monolingual and CS data using curricula C3, C6 and C7 outperforms the CS-data only baseline (C0). Also, C4, that uses alternate sentences of monolingual data outperforms this baseline, probably due to the large difference in data size between the monolingual and CS data.

Although the PPL values for the RNNLM are very high, one must note that the test set consists entirely of CS sentences, whereas the amount of "in-domain" (i.e., CS) data used in training is very low (0.2%). To improve our models, we build the best performing model, the one corresponding to C6, with multiple iterations; the PPL value for it is

[3]Since ngrams models strongly depend on frequency of occurrence, we represent this using a slight abuse of notation to indicate 30 replications of the T_{12} set

Curriculum	RNNLM	Intpl.
C0: T_{12}	27443	477
C1: $T_1; T_2$	46628	1120
C2: $T_{12}; T_1; T_2$	44516	609
C3: $T_1; T_2; T_{12}$	7987	358
C4: $\{T_1, T_2\}$	9749	771
C6: $\{T_1, T_2\}; T_{12}$	**6533** (4544)	**320** (298)
C7: $\{T_1, T_2, T_{12}\}$	23384	673

Table 4: Perplexity results for RNNLM and interpolated RNNLM+ngram LM. Values in parenthesis are for the multiple iteration model.

shown within parenthesis.

Table 4 (column 2) also shows the PPL for models obtained by interpolating the probabilities given by RNNLM and the 5-gram LM. The interpolation coefficient is kept fixed at 0.5. In all cases, the PPL of the ngram model is significantly lower than the RNNLM. For C1/C4, the 5-gram PPL is 915, for C2/C3/C6 it is 778 and for C7 it is 574. However, in all cases, interpolating the ngram model with the RNNLM gives the best results. Experimenting with the interpolation coefficient could potentially give better results.

5.5 Related Work on LM

Language Models for CS have been studied in the context of three main approaches: (a) Bilingual models that combine data from both languages, (b) Factored models that take into account strong indicators of CS like POS and LID, and (c) Models that incorporate linguistic constraints for CS. Bilingual language models are typically trained using pooled text data (Weng et al., 1997). Gebhardt (2011) describes a framework to use Factored Language Models for rescoring n-best lists during decoding. The factors used include POS tags, CS point probability and LID.

In Adel et al.(2014b; 2014a; 2013) recurrent language models built on the same corpus are combined with n-gram based models, or converted to backoff models, giving improvements in perplexity and mixed error rate. Li and Fung (2014) integrates Functional Head constraints for code-switching into the Language Model for decoding a Mandarin-English corpus. Li and Fung (2013) use inversion constraints to predict CS points and integrates this prediction into the decoding process.

Our work is similar to the bilingual model approach in that we pool data from both languages. However, we also add a very small amount of CS data to our models in some of the experiments. In addition, we also focus on the ordering of the monolingual and CS data, which to our knowledge, none of the previous approaches do.

6 Discussion and Conclusion

The experiments presented here on the two tasks and three different DNN architectures show that across all these cases, C6: $\{T_1, T_2\}; T_{12}$ is the most effective curriculum for training CS models. C7: $\{T_1, T_2, T_{12}\}$ also performs quite well, and in absence of CS data, C4: $\{T_1, T_2\}$ seems to be the most effective curriculum. Presenting the monolingual training instances in blocks produce the worst models, and neither is training with the CS data in the beginning any more effective than not using CS data at all.

This is similar to the findings reported in (Shi et al., 2015) in the context of RNNLM adaptation to specific domains. In general, empirical results on transfer learning of DNNs show that training with in-domain data at the end leads to better models. This explains why training first with one language and then another is not ideal; in such cases the model adapts to the second language, as observed for C1, C2 and C3 in Table 2. For similar reasons, training with T_{12} at the beginning provides no benefit. One could possibly argue that during training with $\{T_1, T_2\}$ the upper layers (closer to input) of the networks learns the low level features of the languages. Then during the last phase of training with T_{12}, the lower layers of the network learns when to switch from one language to another, i.e., the CS specific features.

An interesting cognitive metaphor (albeit not an explanation) is as follows: an ideal bilingual is exposed to both the languages almost simultaneously; such a user is able to code-switch even if never exposed to CS (as in C4); however, with exposure to CS, the frequency and perceived-naturalness of CS increases in his/her language use (as in C6). Of course, in multilingual communities children grow up with both languages as well as CS between them (similar to C7). Thus, it is tempting to predict that with large amount of CS training data, i.e., when $|T_{12}| \approx |T_1| \approx |T_2|$, C6 and C7 should perform equally well.

Here, we have explored the ordering of the training instances based on the language. There are other dimensions of complexity, for example the number of code-switch points, syntactic

structure, etc., which could as well be harnessed for more effective curricula. Going forward, we would also like to explore techniques such as Self-paced learning (Kumar et al., 2010; Jiang et al., 2015; Graves et al., 2017).

References

Heike Adel, K. Kirchhoff, N. T. Vu, D. Telaar, and T. Schultz. 2014a. Combining recurrent neural networks and factored language models during decoding of code-switching speech. In *INTERSPEECH*. pages 1415–1419.

Heike Adel, K Kirchhoff, N T Vu, D Telaar, and T Schultz. 2014b. Comparing approaches to convert recurrent neural networks into backoff language models for efficient decoding. In *INTERSPEECH*. pages 651–655.

Heike Adel, N T Vu, and T Schultz. 2013. Combination of recurrent neural networks and factored language models for code-switching language modeling. In *ACL (2)*. pages 206–211.

Peter Auer. 1995. The pragmatics of code-switching: a sequential approach. In L. Milroy and P. Muysken, editors, *One speaker, two languages*, Cambridge Univ Press, pages 115–135.

Kalika Bali, Y. Vyas, J. Sharma, and M. Choudhury. 2014. "I am borrowing ya mixing?" An analysis of English-Hindi code mixing in Facebook. In *Proc. First Workshop on Computational Approaches to Code Switching, EMNLP*.

Yoshua Bengio, J. Louradour, R. Collobert, and J. Weston. 2009. Curriculum learning. In *The 26th annual international conference on machine learning*. ACM, pages 41–48.

Katja F Cantone. 2007. *Code-switching in bilingual children*, volume 296. Springer.

Joyce YC Chan, H. Cao, PC Ching, and T. Lee. 2009. Automatic recognition of cantonese-english code-mixing speech. *Computational Linguistics and Chinese Language Processing* 14(3):281–304.

Joseph Chee Chang and Chu-Cheng Lin. 2014. Recurrent-neural-network for language detection on twitter code-switching corpus. *CoRR* abs/1412.4314.

Amitava Das. 2016. Tool contest on POS tagging for Code-mixed Indian Social Media Text. In *ICON*.

Mona Diab, P. Fung, M. Ghoneim, J. Hirschberg, and T. Solorio, editors. 2016. *Proc.of the 2nd Workshop on Computational Approaches to Code Switching*.

Mona Diab, J. Hirschberg, P. Fung, and T. Solorio, editors. 2014. *Proc. of the 1st Workshop on Computational Approaches to Code Switching*. ACL.

Margreet Dorleijn. 2016. Can internet data help to uncover developing preferred multilingual usage patterns? an exploration of data from turkish-dutch bilingual internet fora. *Journal of Language Contact* 9(1):130–162.

Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition* 48(1):71–99.

Akshay Gadre, R. Begum, K. Bali, and M. Choudhury. 2016. Machine translating code mixed text: Pain points and sweet spots. In *WILDRE*.

Jan Gebhardt. 2011. Speech recognition on english-mandarin code-switching data using factored language models .

Javier Gonzalez-Dominguez, D Eustis, I Lopez-Moreno, A Senior, F Beaufays, and Pedro J Moreno. 2015. A real-time end-to-end multilingual speech recognition architecture. *IEEE Journal of Selected Topics in Signal Processing* 9(4):749–759.

Alex Graves, M G Bellemare, J Menick, R Munos, and K Kavukcuoglu. 2017. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003* .

Aaron Jaech, G Mulcaire, S Hathi, M Ostendorf, and N A Smith. 2016. A neural model for language identification in code-switched tweets. *EMNLP 2016* page 60.

Anupam Jamatia and A Das. 2014. Part-of-speech tagging system for Hindi social media text on twitter. In *The First Workshop on Language Technologies for Indian Social Media, ICON*.

Anupam Jamatia, B Gambck, and A Das. 2015. Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages. In *RANLP*.

Lu Jiang, D Meng, Q Zhao, S Shan, and A G Hauptmann. 2015. Self-paced curriculum learning. In *AAAI*. volume 2, page 6.

Melvin Johnson, M Schuster, Q V Le, M Krikun, Y Wu, Z Chen, N Thorat, F Viégas, M Wattenberg, G Corrado, et al. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv preprint arXiv:1611.04558* .

M Pawan Kumar, B Packer, and D Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*. pages 1189–1197.

Ying Li and P Fung. 2013. Improved mixed language speech recognition using asymmetric acoustic model and language model with code-switch inversion constraints. In *ICASSP*. pages 7368–7372.

Ying Li and P Fung. 2014. Language modeling with functional head constraint for code switching speech recognition. In *EMNLP*.

Dau-Cheng Lyu, R-Y Lyu, Y-C Chiang, and C-N Hsu. 2006. Speech recognition on code-switching among the chinese dialects. In *IEEE ICASSP 2006*. volume 1, pages I–I.

Tetyana Lyudovyk and Valeriy Pylypenko. 2014. Code-switching speech recognition for closely related languages. In *SLTU*. pages 188–193.

Tomas Mikolov, S Kombrink, A Deoras, L Burget, and J Cernocky. 2011. RNNLM-recurrent neural network language modeling toolkit. In *ASRU Workshop 2011*. pages 196–201.

Giovanni Molina, N Rey-Villamizar, T Solorio, F AlGhamdi, M Ghoneim, A Hawwari, and M Diab. 2016. Overview for the second shared task on language identification in code-switched data. *EMNLP 2016* page 40.

Carol Myers-Scotton. 1993. *Dueling Languages: Grammatical Structure in Code-Switching*. Claredon, Oxford.

Shruti Rijhwani, R Sequiera, M Choudhury, K Bali, and C S Maddila. 2017. Estimating code-switching on Twitter with a novel generalized word-level language identification technique. In *ACL*.

Koustav Rudra, S Rijhwani, R Begum, K Bali, M Choudhury, and N Ganguly. 2016. Understanding language preference for expression of opinion and sentiment: What do Hindi-English speakers do on Twitter? In *EMNLP*. pages 1131–1141.

Younes Samih, S Maharjan, M Attia, L Kallmeyer, and T Solorio. 2016. Multilingual code-switching identification via lstm recurrent neural networks. In *2nd Workshop on Computational Approaches to Code Switching,*. pages 50–59.

Royal Sequiera, M Choudhury, and K Bali. 2015a. Pos tagging of Hindi-English code mixed text from social media: Some machine learning experiments. In *Proceedings of ICON*.

Royal Sequiera et al. 2015b. Overview of fire-2015 shared task on mixed script information retrieval. In *FIRE*. pages 21–27.

Shashank Sharma, P Srinivas, and R C Balabantaray. 2015. Text normalization of code mix and sentiment analysis. In *ICACCI, 2015 International Conference on*. IEEE, pages 1468–1473.

Yangyang Shi, M Larson, and C M Jonker. 2015. Recurrent neural network language model adaptation with curriculum learning. *Computer Speech & Language* 33(1):136–154.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for english-spanish code-switched text. In *Proc. of EMNLP*.

Thamar Solorio et al. 2014. Overview for the first shared task on language identification in code-switched data. *In 1st Workshop on Computational Approaches to Code Switching, EMNLP* pages 62–72.

Valentin I Spitkovsky, H Alshawi, and D Jurafsky. 2009. Baby steps: How less is more in unsupervised dependency parsing. *NIPS: Grammar Induction, Representation of Language and Language Learning* pages 1–10.

Andreas Stolcke et al. 2002. SRILM-an extensible language modeling toolkit. In *Interspeech*. volume 2002, page 2002.

Yogarshi Vyas, S Gella, J Sharma, K Bali, and M Choudhury. 2014. POS Tagging of English-Hindi Code-Mixed Social Media Content. In *Proc. EMNLP*. pages 974–979.

Fuliang Weng, H Bratt, L Neumeyer, and A Stolcke. 1997. A study of multilingual speech recognition. In *EUROSPEECH*. Citeseer, volume 1997, pages 359–362.

Dong Yu, A Eversole, M Seltzer, K Yao, Z Huang, B Guenter, O Kuchaiev, Y Zhang, F Seide, H Wang, et al. 2014. An introduction to computational networks and the computational network toolkit. *Microsoft Technical Report MSR-TR-2014–112* .

Quantitative Characterization of Code Switching Patterns in Complex Multi-Party Conversations: A Case Study on Hindi Movie Scripts

Adithya Pratapa
Microsoft Research, India
adithyapratapa@gmail.com

Monojit Choudhury
Microsoft Research, India
monojitc@microsoft.com

Abstract

In this paper, we present a framework for quantitative characterization of code-switching patterns in multi-party conversations, which allows us to compare and contrast the socio-cultural and functional aspects of code-switching within a set of cultural contexts. Our method applies some of the proposed metrics for quantification of code-switching (Gamback and Das, 2016; Guzman et al., 2017) at the level of entire conversations, dyads and participants. We apply this technique to analyze the conversations from 18 recent Hindi movies. In the process, we are able to tease apart the use of code-switching as a device for establishing identity, socio-cultural contexts of the characters and the events in a movie.

1 Introduction

Code-switching (henceforth CS) or *code-mixing* refers to the juxtaposition of linguistic units from more than one language in a single conversation, or in a single utterance. Linguists have extensively studied the structural (i.e., the grammatical constraints on CS) and functional (i.e., the motivation and intention behind CS) aspects of CS in various mediums, contexts, languages and geographies (Myers-Scotton, 2005; Auer, 1995, 2013). However, most of these studies are limited to qualitative analysis of small datasets, which makes it hard to make statistically valid quantitative claims over the nature and distribution of CS.

Recently, due to the availability of large code-switched datasets, gathered mostly from social media, there has been some quantitative studies on socio-linguistic and functional aspects of CS (Rudra et al., 2016; Rijhwani et al., 2017; Guzman et al., 2017). Nevertheless, there are no large-scale quantitative studies of code-switched conversations, primarily because currently the only available large-scale datasets come from social media. These are either micro-blogs without any conversational context or data from Facebook or WhatsApp with very short conversations. On the other hand, functions of CS are most relevant and discernible in relatively long multi-party conversations embedded in a social context. For instance, it is well documented (Auer, 2013) that CS is motivated by complex social functions, such as identity, social power and style accommodation, which are difficult to elicit and establish from short social media texts.

In this work, we propose a set of techniques for analyzing CS styles and functions in conversations grounded over social networks. Our approach develops on two previously proposed metrics of CS – the *Code-mixing Index* (CMI) (Gamback and Das, 2016) and corpus level metrics proposed in (Guzman et al., 2017), applied to conversations at the level of dyads, participants, conversation scenes and the entire social network of the participants. We apply this new approach to analyze scripts of 18 recent Hindi movies with various degrees and styles of Hindi-English CS. Through this analysis technique, we are able to bring out the social functions of CS at different levels.

The primary contributions of this work are: (a) development of a set of quantitative conversation analysis techniques for CS; (b) some visualization techniques for CS patterns in conversations that can help linguists and social scientists to get a holistic view of the switching styles in interactions; (c) analysis of CS patterns in recent Hindi movies that adds to the existing rich literature of similar but small scale qualitative studies of CS in Indian cinema.

Rest of this paper is organized as follows: Sec

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 75–84

2 describes related work on functions of CS with particular emphasis on CS in Indian cinema. Sec 3 introduces our analysis technique, which is later applied and illustrated in the context of movie scripts in Sec 5 and 6. Sec 4 introduces the movie dataset, preprocessing of the scripts and word-level language labeling of the dialogues. Sec 7 concludes the paper by summarizing the contributions and discussing potential future work.

2 Related Work

In this section, we will start with a brief review of the linguistics literature on functional and socio-linguistic aspects of CS, followed by a discussion on recent computational models. In order to put the case-study on Hindi movies in perspective, we will also review relevant literature on CS in Indian cinema.

2.1 Functions of Code-Switching

Code-switching is a common phenomenon in all multilingual communities, though usually it is unpredictable whether in a given context a speaker will code-switch or not (Auer, 1995). Nevertheless, linguists have observed that there are preferred languages for communicating certain kinds of functions. For instance, certain speech activities might be exclusively or more commonly related to a certain language choice (e.g. Fishman (1971) reports use of English for professional purposes and Spanish for informal chat for English-Spanish bilinguals from Puerto Rico). Language switching is also used as a signaling device that serves specific communicative functions (Barredo, 1997; Sanchez, 1983; Nishimura, 1995; Maschler, 1991, 1994) such as: (a) reported speech (b) narrative to evaluative switch (c) reiterations or emphasis (d) topic shift (e) puns and language play (f) topic/comment structuring etc. Attempts of predicting the preferred language, or even exhaustively listing such functions, have failed. However, linguists agree that language alteration in multilingual communities is not a random process.

Code-switching is also strongly linked to social identity and the principle of linguistic style accommodation (Melhim and Rahman, 1991; Auer, 2013). For instance, two Hindi-English bilingual speakers could code-switch just to establish a connection or in-group identity because CS is the norm for a large section of urban Indians, and English is attached to aspirational values by a large section of the Indian society (see Sec.2.3 for detailed discussion on this).

2.2 Computational and Quantitative Studies

Over the last decade, research in computational processing of code-switching has gained significant interest (Solorio and Liu, 2008, 2010; Vyas et al., 2014; Peng et al., 2014; Sharma et al., 2016). In particular, word-level language identification, which is the first step towards processing of CS text, has received a lot of attention (see Rijhwani et al. (2017) for a review). In this work, we use the word-level language labeler by Gella et al. (2013) for labeling the Hindi movie dialogues.

Nevertheless, to the best of our knowledge, there has been very little work on automatic identification of functional aspects of CS or any large-scale data-driven study of its socio-linguistic aspects. Of the few studies that exist, most notable are the ones by Rudra et al. (2016) on language preference by Hindi-English bilinguals on Twitter and Rijhwani et al. (2017) on extent and patterns of CS across European languages from 24 cities. Rudra et al. (2016) analyzed 430K unique tweets for opinion and sentiment, and concluded that Hindi-English bilinguals prefer to express negative opinions in Hindi; they further report that a large fraction of the CS tweets exhibited the narrative-evaluative function. Rijhwani et al. (2017) examined more than 50M tweets from across the world the study shows that the percentage of CS tweets varies from 1 to 11% across the cities, and more CS is observed in the cities where English is not the primary language of communication. They also show that English-Spanish CS patterns in a predominantly Spanish speaking region (e.g., Barcelona) are different from those where English is the primary language (e.g., Houston).

In an excellent survey on computational socio-linguistics, Nguyen et al. (2016) report a few other studies on socio-linguistic aspects of multilingual communities.

2.3 Code-switching in Indian Cinema

Hindi-English CS, commonly called *Hinglish*, is extremely widespread in India. There is historical attestation, as well as recent studies on the growing use of Hinglish in general conversation, and in entertainment and media (see Parshad et al. (2016) and references therein). Several recent studies (Bali et al., 2014; Barman et al., 2014;

Sequiera et al., 2015) also provide evidence of Hinglish and other instances of CS on online social media, such as Twitter and Facebook.

Hindi movies provide a rich data source for studying CS in the Indian context. According to the *Conversational Analysis* approach to CS (Auer, 2013; Wei, 2002), in any given context a particular language is preferred or *unmarked*. Therefore, "speakers, and in turn script writers, choose marked or unmarked codes on the basis of which one will bring them the best outcomes" (Vaish, 2011). Myers-Scotton (2005) suggested that the *matrix* or unmarked code for Hindi movies is Hindi. Therefore, any switch to English has some communicative purpose. Lösch (2007) uses this idea to analyze the dialogues of the movie Monsoon Wedding (2001) and concludes that English is used as a device for encoding social distance; lower socio-economic class characters switch to English for upward social mobility.

Vaish (2011), on the other hand, argues that Hindi is not necessarily the matrix or the unmarked code for all characters and scenes in current Hindi movies. Instead, the two codes (and sometimes even more languages and regional varieties) are used to bring out the identity of each character. In particular, English and Hinglish are associated with Westernization of culture, and are often used as the preferred code for depicting NRI or otherwise strongly westernized characters in the movies. Yet a third line of study by Kachru (2006) argues that predominance of English in Hindi movies crops from the fact that it helps the screenplay writers to borrow fresh metaphors and new rhyming words from English; it also adds to the playfulness, irony, humor and satire.

Chandra et al. (2016) report an acute rise in use of English words in Hindi song lyrics over the years. This is the only quantitative study of CS in Indian cinema that we are aware of.

3 Approach

In this section, we present the techniques that can be used to study complex multi-party conversations like plays, movies, Facebook/WhatsApp group conversations, and so on. We propose a domain independent modular framework to quantitatively analyze these conversations. For this, we adopt metrics proposed by Guzman et al. (2017) and Gamback and Das (2016) to comprehensively measure various aspects of CS in the corpus.

3.1 Metrics for Quantification of CS

The first corpus level quantification of the extent and nature of CS was proposed by Gamback and Das (2016). Referred to as the **Code mixing index**, this metric tries to capture the language distribution and the switching, both at the level of utterances and the entire corpus. Let N be the number of languages, x an utterance; let t_{L_i} be the tokens in language L_i, P be the number of code alternation points in x; also, let w_m and w_p be the weights for the two components of the metric. Then, the *Code mixed index per utterance*, $C_u(x)$ for x is:

$$C_u(x) = 100\frac{w_m(N(x) - \max_{L_i \in L}\{t_{L_i}\}(x)) + w_p P(x)}{N(x)} \quad (1)$$

Let U be the number of utterances in the corpus and $S \leq U$ be the number of utterances that contains code-switching. Then the *Code mixed index over the entire corpus*, C_c is defined as:

$$C_c = \frac{\sum_{x=1}^{U} C_u(x) + w_p\delta(x)}{U} + w_s.\frac{S}{U}.100 \quad (2)$$

$$\delta(x) = \begin{cases} 0, & x = 1 \vee L_{x-1} = L_x \\ 1, & x \neq 1 \wedge L_{x-1} \neq L_x \end{cases} \quad (3)$$

In another recent study, Guzman et al. (2017) propose not a single, but rather a set of metrics for quantification of CS in a corpus. These are:

M-Index captures the inequality of distribution of languages in the corpus. Let p_j be the fraction of words in language j and k represents the total number of languages in the corpus, then

$$\textit{M-index} = \frac{1 - \sum p_j^2}{(k-1)\sum p_j^2} \quad (4)$$

Language Entropy is the number of bits needed to represent the distribution of languages.

$$\textit{LE} = -\sum_{j=1}^{k} p_j log_2(p_j) \quad (5)$$

I-Index is the switching probability.

$$\textit{I-index} = \frac{\text{Total no. of switch points}}{n-1} \quad (6)$$

Burstiness quantifies whether the switching has periodic character or occurs in bursts. Let σ_τ, m_τ be the standard deviation and the mean of language-span (in terms of number of words in a

contiguous sequence of words in a language) distributions respectively.

$$\textit{Burstiness} = \frac{\sigma_\tau - m_\tau}{\sigma_\tau + m_\tau} \tag{7}$$

Span Entropy (SE) is the number of bits needed to represent the distribution of language spans. If p_l represents sample probability of a span of length l, then

$$\textit{SE} = -\sum_{l=1}^{M} p_l log_2(p_l) \tag{8}$$

Memory captures the tendency of consecutive language spans to be positively or negatively autocorrelated. n_r is the number of language spans in the distribution, τ_i is the language span under consideration, σ_1 and m_1 are the standard deviation and the mean of all spans except the last, whereas σ_2 and m_2 are the standard deviation and the mean of all spans except the first,

$$\textit{Memory} = \frac{1}{n_r - 1} \sum_{i=1}^{n_{r-1}} \frac{(\tau_i - m_1)(\tau_{i+1} - m_2)}{\sigma_1 \sigma_2} \tag{9}$$

Each of these metrics evaluate a different aspect of the corpus. For example, M-Index captures the multilingualism of the corpus whereas CMI can be used to measure the switching between languages in and across the utterances. Therefore, an analytical approach that combines all these metrics and overlays it on top of the conversation network of the participants can bring out the various social and functional aspects of CS.

3.2 The Proposed Approach

Here, we present a systematic approach to analyze CS conversations. We begin with a set of definitions and notations. Though the concepts defined below applies to any multi-party conversation, it might be useful to think of these in the context of a play or a movie.

Let $\mathbf{P} = \{P_1, P_2, \ldots P_k\}$ represents a set of *participants* (akin to characters in a play or movie). Let us define a conversation *scene* S_i as a sequence of *participant-utterance* pairs: $\{\langle P_{1,i}, U_{1,i}\rangle, \langle P_{2,i}, U_{2,i}\rangle, \ldots \langle P_{m_i,i}, U_{m_i,i}\rangle\}$. This is essentially a multi-party conversation where each participant $P_{j,i} \in \mathbf{P}$ speaks out $U_{j,i}$ during the conversation. Finally, a series of such *scenes*, $\{S_1, S_2, \ldots\ldots, S_n\}$ among the participants in $\mathbf{P}$ along with their social context constitute a *socially grounded multi-party, multi-scene conversational corpus*, which we shall simply refer here as the corpus[1] $\mathbf{C}$. Thus, $\mathbf{C}$ is similar to the script of an entire movie or a play.

Note that while the social context of a scene, such as the presence of passive participants, the occassion and location, etc., are extremely important for understanding the CS patterns, in the current study we will ignore these meta-variables altogether. Our analysis will solely rely on computing the CS metrics on the set of utterances present in the entire corpus, which we shall denote as $\pi(\mathbf{C})$. Here, π refers to a projection of all the utterances present in $\mathbf{C}$.

Further, this projection can be limited to scenes, participants, or dyads, which are defined below.

- $\pi_{P_i}(\mathbf{C}) \rightarrow$ set of all the utterances of the participant $P_i \in \mathbf{P}$ in $\mathbf{C}$
- $\pi_{S_j}(\mathbf{C}) \rightarrow$ set of all the utterances in the scene S_j in $\mathbf{C}$
- $\pi_{D_{i,j}}(\mathbf{C}) \rightarrow$ set of all the utterances of the dyad (P_i, P_j), $P_i, P_j \in \mathbf{P}$ in $\mathbf{C}$. A *dyad* is defined as two consecutive utterances in any *scene*, where the first and the second participant are P_i and P_j, not necessarily in that order.

The metrics described in the earlier subsections can be applied to any of these projections and they can be separately analyzed for inferences. We propose three kinds of analysis,

- **Corpus**: We can visualize each corpus $\mathbf{C}$ based on these metrics and a cross-corpus comparison can be made to explain the sociocultural setting of each of the corpora (or movie).
- **Participant**: We can visualize the metrics for a participant over the entire corpora and a cross-participant comparison can reveal patterns relating the social identity of the participants.
- **Dyad**: Similar analysis can be done for each dyad and this can help us find the functional reasons for code switching, for example trying to accommodate the other participant in the conversation.

[1]Note that a collection of movie scripts, such as the one analyzed here would usually be referred to as a corpus. However, here, we will refer to each movie as a conversational corpus.

- **Conversation Network:** We can overlay the cross-metric comparison plots onto the network graph of the participants and this allows us to study the variations in the amount and style of CS by a participant with the other participants in the network.

Thus, we can see the wide range of insights this line of analysis could provide, and in the next three sections we will illustrate these techniques through a case study on movies.

4 Dataset

Though our methodology can be applied to any complex multi-party conversation, in this work we apply our framework to the case of Hindi films.

For our study we chose 18 recent Hindi film scripts from a blog (https://moifightclub.com/category/scripts/), which has around two dozen Hindi movie scripts. The movies with their meta-data and basic corpus statistics are presented in Table 1.

We processed the scripts from the above blog in the following way, (i) Converted the scripts pdfs to text (ii) Using simple regular expressions, we extracted the characters, dialogues and also segregated the script into scenes (iii) Language labeled the dialogue using the tagger developed by (Gella et al., 2013) into one of Hi (Hindi), En (English) and Other. The language tagger uses context switch probability and monolingual frequency factor on the top of maximum entropy classifier to classify the Hi-En data.

A dialogue snippet from the script of movie *Queen* is shown below. All the English words are italicized and loose literal translations in English are given within angular brackets. As we can see both intra-sentential and inter-sentential CS is present in this snippet.

The distribution of the languages are presented in Table 1 and we see significant usage of English in all movies. Overall, we have noticed four kinds of errors in processing the scripts. First being the limitations of pdf to text converter, where formating and justification issues lead to word splitting, but these are very few in the corpus. Second, we initially missed out the dialogues that were capitalized. All the characters in the scripts are in caps and our cues are built accordingly. We tried to minimize these errors by manually identifying the characters after preprocessing. An example of the third kind of error is, some characters like 'Vijaylaxmi' in the movie Queen are initially represented by generic phrases like 'The French Girl' before the character introduces itself. These errors are also few and in general there are very few dialogues by the character before his/her introduction. Lastly, the errors caused by language tagger and we observed the accuracy to be slightly lower than the results presented in the original paper.

VIJAY:
ek minute ke liye thoda *practical* socho
⟨ Please think practically for a minute ⟩
VIJAY:
Main tumharey *angle* se hi soch raha hoon... Tum hi *uncomfortable feel* karogi... bahut *time* ho gaya hai... bahut fark aa gaya hai
⟨ I am thinking from your perspective... But you will feel uncomfortable.. long time has passed.. things have changed a lot ⟩
RANI:
Kismein? Mujhmein koyi *change* nahin hai
⟨ In whom? I haven't changed at all ⟩
VIJAY:
Vohi to baat hai... mujhmein hai... meri duniya ... bilkul alag hai... ab... *you'll not fit in*
⟨ That's the point... I have.. My world... is very different... now... you'll not fit in ⟩
RANI:
Matlab? ek dum se main tumharey jitni *fancy* nahin hoon...
⟨ What do you mean? Suddenly I am no longer as fancy as you ⟩

The preprocessed corpus is available for research on request by email to the authors.

5 Corpus level Analysis

In this section we present the results of the metrics discussed in section 3 on the entire corpus. The results of the metrics are given in table 2 and are indexed by the Movie ID (as in table 1). The table presents the metrics detailed in the section 3.1 with the first half being the ones proposed by (Guzman et al., 2017) and the later by (Gamback and Das,

MID	Movie (Year)	Script Writer	Director	% HI	% EN	# words	# turns
1	Ankhon dekhi (2014)	Rajat Kapoor	Rajat Kapoor	69.66	17.27	11940	753
2	D-day (2013)	Nikhil Advani et al,	Nikhil Advani	62.95	21.46	10904	659
3	Dedh ishqiya (2014)	Vishal Bhardwaj et al.	Abhishek Chaubey	68.81	14.74	7775	642
4	Dum laga ke haisha (2015)	Sharat Katariya	Sharat Katariya	67.03	15.52	8870	678
5	Ek main aur ek tu (2012)	Ayesha Devitre, Shakun Batra	Shakun Batra	39.53	42.35	10333	836
6	Kapoor and sons (2016)	Shakun Batra, Ayesha D. Dillion	Shakun Batra	49.72	32.36	13698	1119
7	Kai po che (2013)	Pubali Chaudhari et al.	Abhishek Kapoor	56.83	26.79	11670	675
8	Lootera (2013)	Bhavani Iyer, V. Motwane	V. Motwane	71.4	12.7	8314	734
9	Masaan (2015)	Varun Grover	Neeraj Ghaywan	59.78	20.83	7620	653
10	Neerja (2016)	Saiwyn Quadras	Ram Madhvani	53.47	32.63	8293	602
11	NH10 (2015)	Sudip Sharma	Navdeep Singh	34.43	42.53	3148	340
12	Pink (2016)	Shoojit Sricar et al.	A. Roy Chowdhury	46.39	39.69	15437	897
13	Queen (2014)	Vikas Bahl et al.	Vikas Bahl	47.6	35.51	8958	951
14	Raman Raghavan 2.0 (2016)	Anurag Kashyap, Vasan Bala	Anurag Kashyap	63.35	20.42	5171	373
15	Shahid (2013)	Sameer Gautam Singh	Hansal Mehta	47.47	34.17	10084	896
16	Talvar (2015)	Vishal Bhardwaj	Meghna Gulzar	48.97	34.9	9957	823
17	Titli (2015)	Sharat Katariya, Kanu Behl	Kanu Behl	49.01	34.7	8368	656
18	Udaan (2010)	V. Motwane, Anurag Kashyap	V. Motwane	64.53	18.59	10545	955

Table 1: List of Movies analyzed with some basic statistics. MID - Movie Id.

2016). $\mathbf{C_c}$ represents the CMI values on the overall corpus while $\mathbf{C_u}$ **mix** and $\mathbf{C_u}$ **total** denote the CMI per utterance averaged over the mixed and total utterances respectively. **P mix** and **P total** are the average number of switch points in the set of mixed and total utterances respectively. The last two columns represent the number and percentage of inter-switches (change of matrix language) in the corpus. We can see a significant variation of most of the metrics across the movies. Figure 1 shows the distribution of mixed and non-mixed utterances for the movies and this captures the mixing in dialogues in contrast to the switching in the entire corpus. On average, 50% of the dialogues in a movie are code mixed and signifies the use of multilingualism in the movie corpus.

Figure 2 represents the movies in an M-index vs CMI scatter plot ($\pi(C)$). As shown, the movies can be visually clustered into three sets: Cluster **A** has movies with low CS (both low CMI and M-Index), cluster **B** has movies with high CS (both high CMI and M-Index), and cluster **C** contains movies that has high M-Index (approximately equal usage of Hi and En) but low CMI. Each of these clusters can be explained based on the socio-cultural setting of the movies. For instance, the movies in cluster **B** are based in urban setting and have more CS than the movies in cluster **A**, which are typically based in small towns (e.g., Dum laga ke haisha), rural settings (e.g., Udaan), or in the past (e.g., Lootera). On the other hand, the movies in cluster **C** like Queen are the ones away from the trend-line (shown as the dotted line) and it is because they have different matrix languages for different parts of the movie. This results in an overall high M-Index value but there is very little code switching in the scenes with English as matrix language, leading to lower CMI. We also compared other metrics but gained very similar insights.

MID	M-metric	I-metric	Bursti-ness	Memory	Language Entropy	Span Entropy	CMI Metrics						
							C_c	C_u mix	C_u total	P mix	P total	# IS	% IS
1	0.467	0.221	0.117	-0.203	0.719	3.240	72.47	30.11	19.24	4.16	2.66	146	19.39
2	0.611	0.210	0.081	-0.212	0.818	3.431	81.41	35.63	24.38	3.68	2.52	172	26.1
3	0.410	0.190	0.140	-0.248	0.672	3.480	58.13	32.56	16.33	3.04	1.52	190	29.6
4	0.440	0.194	0.128	-0.253	0.697	3.478	66.09	31.27	18.03	2.9	1.67	157	23.16
5	0.998	0.232	0.062	-0.005	0.999	3.318	77.08	50.63	29.13	3.37	1.94	352	42.11
6	0.914	0.240	0.066	-0.076	0.968	3.263	80.26	47.58	29.17	3.15	1.93	486	43.43
7	0.771	0.236	0.091	-0.131	0.905	3.244	90.99	39.26	29.14	3.97	2.95	198	29.33
8	0.345	0.172	0.139	-0.294	0.612	3.629	52.42	30.83	14.16	2.68	1.23	194	26.43
9	0.621	0.223	0.121	-0.206	0.824	3.287	65.26	37.05	20.09	3.09	1.68	186	28.48
10	0.889	0.197	0.143	-0.152	0.957	3.486	72.42	38.45	22.87	3.24	1.93	164	27.24
11	0.978	0.217	0.210	-0.092	0.992	3.249	54.78	41.67	18.26	2.64	1.16	105	30.88
12	0.988	0.209	0.080	-0.075	0.996	3.468	78.27	47.17	28.29	4.38	2.63	394	43.92
13	0.959	0.216	0.129	-0.144	0.985	3.360	53.47	43.8	18.42	3.04	1.28	353	37.12
14	0.584	0.189	0.103	-0.208	0.801	3.525	62.22	35.69	18.66	3.44	1.8	101	27.08
15	0.948	0.205	0.035	-0.129	0.981	3.509	59.97	47.09	21.65	2.99	1.38	419	46.76
16	0.945	0.249	0.093	-0.067	0.980	3.156	69.43	45.37	24.48	3.82	2.06	352	42.77
17	0.943	0.212	0.018	-0.037	0.979	3.458	76.95	46.1	27.41	3.06	1.82	271	41.31
18	0.532	0.229	0.074	-0.251	0.767	3.283	57.57	38.03	18.04	3.46	1.64	343	35.92

Table 2: Metrics

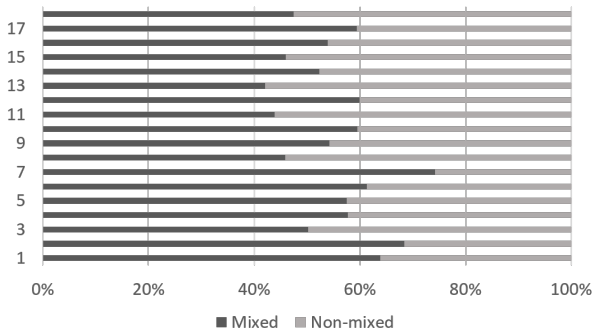


Figure 1: Percentage of Code-switched utterances in the movies.

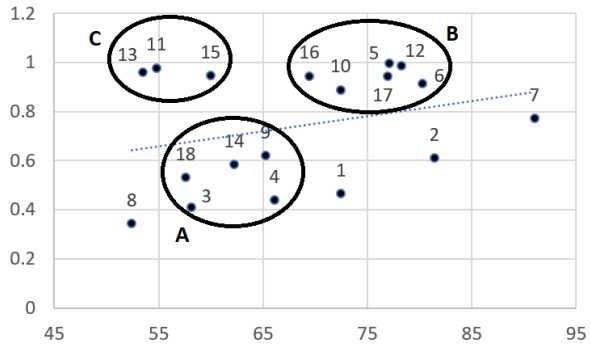


Figure 2: Movies plotted on M-Index (y-axis) vs CMI (x-axis) scatter

6 Participant Level Analysis

In this section we analyze character and dyad specific aspects of CS patterns in the movies. We compute the metrics, M-Index and CMI for corpus projected on participants and dyads. Figure 3 shows the standard deviation of CMI and M-index over all participants and dyads in the movies. The plots indicate that there are significant differences in the patterns across the movies. For instance, MID-13 *Queen* shows large variation in the amount of CS used by the various characters and dyads; whereas, MID-18 *Udaan* has very little variation in the extent of CS exhibited by the characters and dyads. MID-15 *Shahid* shows yet another different pattern, where all characters have similar levels of CS, though there is a larger variation across the dyads. Thus, one can conclude that in *Queen* CS is used to establish the identity of the characters; in *Shahid*, CS is used for establishing the social dynamics of the relations (dyads), but not necessarily the characters; and in *Udaan*, CS is neither used to establish characters or the dyadic relationships; rather in this movie, the CS is used to bring out the overall socio-cultural setting of the movie.

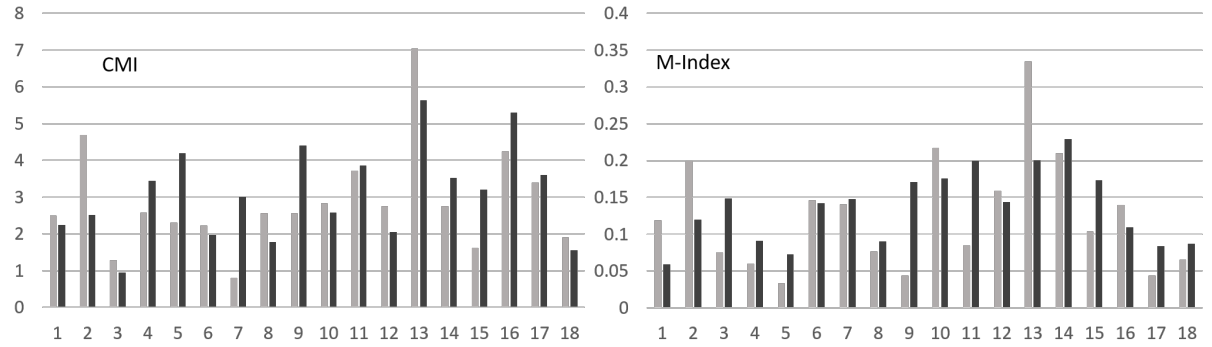


Figure 3: Standard Deviation for characters (light grey) and dyads (dark grey) for CMI (left) and M-Index (right).

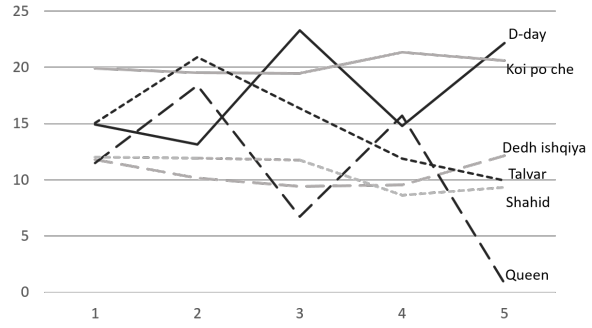


Figure 4: CMI of the the five characters ranked in ascending order of the number of dialogues in the movie, for six movies.

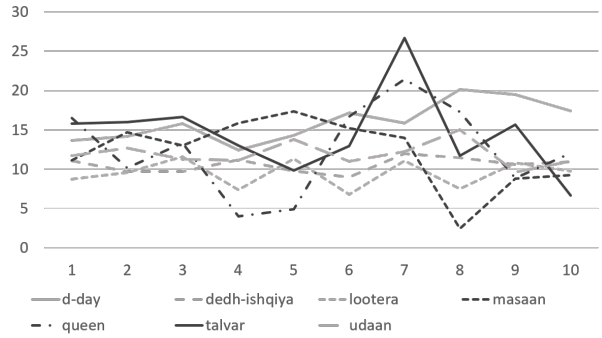


Figure 5: CMI for top 10 dyads for 6 movies.

In order to understand and characterize these differences further, for each movie we ranked the participants/dyads by their utterance count and plot the standard deviation for the top 5 participants and top 10 dyads. Figure 4 and 5 shows these plots, respectively for the characters and the dyads, for the top and bottom three movies in terms of the variance in CS (by CMI).

In the participant plot, Queen, D-day and Talvar are the movies with highest variance while Kai po che, Dedh ishqiya and Shahid are the ones with lowest variance. In the movie Queen, the characters 'Vijaylaxmi' and 'Mikhaelo' exhibit little CS since they speak only or mainly English owing their identity. On the other hand, 'Rani' ,'Vijay' and 'Mom' are based in Delhi, India and they exhibit high CS. Similarly in the case of D-day, the character 'Aslam' has multiple roles in the movie. In order to distinguish between the roles, high CS is used for one of the roles, compared to the other prominent characters. Thus, we observe that CS is used as a tool by the scripts writers to depict the identity of the characters.

In Figure 6 we see that for the movies Queen and Talvar dyads exhibit high variation in CS, whereas in D-day, Udaan, Dedh ishqiya and Lootera there is very little variation across the top 10 dyads. It is interesting to note that for the movie D-day, the characters show low but the dyads show high variation, unlike the movie Queen where the variation is high for both. In order to further investigate these variations, we plotted the character network graphs for these movies on the top of their CMI-M Index plot, also denoting the average M-Index and CMI for the entire movie (figure 6 and 7).

The diameter of the circle denoting the participant $d_{P_i} \propto \sqrt{|\pi_{P_i}(C)|}$ and the thickness and darkness of the edge between two participants are $t_{P_i,P_j} \propto log|\pi_{D_{i,j}}(C)|$ and $d_{P_i,P_j} \propto log|Cc(\pi_{D_{i,j}}(C))|$ respectively.

We observe a clear difference in the networks for Queen and D-day. In the case of Queen, the movie revolves around the central character 'Rani' and all others characters have dialogues primarily with 'Rani'. These characters are from different countries (India, France, Japan, Russia) and the

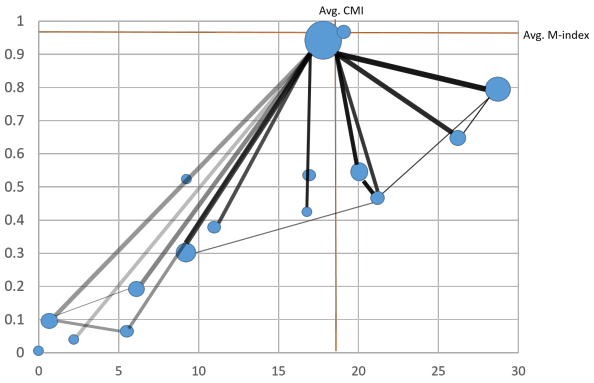


Figure 6: Queen - Network Plot

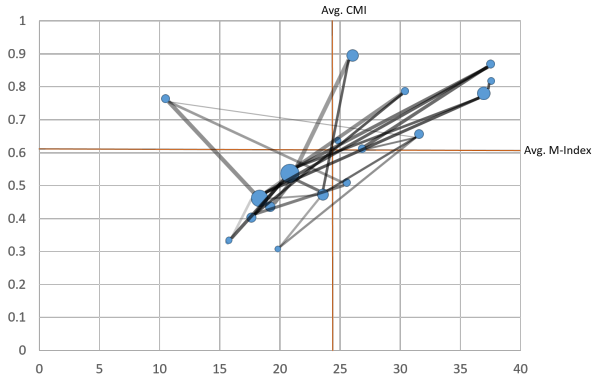


Figure 7: D-day - Network Plot

CS in dialogues with the central character varies a lot, as captured by the darkness of edges in the graph d_{P_i,P_j}. The individual amount of CS also widely varies depending on the country of origin with higher Hi-En CS for characters based in India. The overall mean CMI and M-Index of the movie are closer to the central character as she have many more dialogues than most others. Whereas in D-day the characters are distributed around the movie's average metrics and the graph is well-connected. The CS patterns across the characters and dyads are more similar than in Queen. Thus, in Queen, we see CS being used to represent social identity of the characters but not so much in D-day. As we have already illustrated the socio-cultural context of the movies is also inherently captured by code switching. Due to paucity of space we have only presented our analysis for two movies but we observed similar trends across the movies.

7 Conclusion

In this work, we presented a framework for quantitative characterization of CS patterns in multi-party conversations which goes beyond the existing techniques of corpus level footprints. We apply this approach to analyze scripts of 18 Hindi movies and illustrate its effectiveness in bringing out certain social aspects of CS, such as establishment of identity. Our study also reveals the widely different styles and frequency in which CS is employed as a strategy to establish identity and social context in the movies.

We would like to emphasize that the approach presented here can be extended in scope as well as applied to a wide genre of conversational data, including but not limited to, social media text, private and group chat (e.g., Whatsapp), trascribed speech corpora and literary work. In terms of scope, the approach can be used to study linguistic style accomodation with respect to CS, and pragmatic functions and structural aspects of code-switching.

Acknowledgement

We would like to thank Anupam Jamatia, Amitava Das and Bjorn Gamback for providing us with the code for computing CMI metrics, and Gualberto Guzman, Barbara Bullock and Jaqueline Toribio for sharing the code for computing other set of metrics. We would also like to thank Kalika Bali and Sunayana Sitaram for their insights and valuable inputs on this work. We thank the authors of the blog for allowing us to use the scripts for our research.

References

Peter Auer. 1995. The pragmatics of code-switching: a sequential approach. In Lesley Milroy and Pieter Muysken, editors, *One speaker, two languages*, Cambridge University Press, pages 115–135.

Peter Auer. 2013. *Code-switching in conversation: Language, interaction and identity*. Routledge.

Kalika Bali, Yogarshi Vyas, Jatin Sharma, and Monojit Choudhury. 2014. "I am borrowing ya mixing?" an analysis of English-Hindi code mixing in Facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

Inma Muñoa Barredo. 1997. Pragmatic functions of code-switching among Basque-Spanish bilinguals. *Retrieved on October* 26:528–541.

Subhash Chandra, Bhupendra Kumar, Vivek Kumar, and Sakshi. 2016. Acute sporadic english in bollywood film songs lyrics: A textual evidence based analysis of code-mixing in hindi. *Language in India* 16(11):25–34.

J. A. Fishman. 1971. *Sociolinguistics*. Rowley, Newbury, MA.

B. Gamback and A Das. 2016. Comparing the level of code-switching in corpora. In *Proc. of the 10th International Conference on Language Resources and Evaluation (LREC)*.

Spandana Gella, Jatin Sharma, and Kalika Bali. 2013. Query word labeling and back transliteration for indian languages: Shared task system description .

Gualberto Guzman, Joseph Ricard, Jacqueline Serigos, Barbara E. Bullock, and Almeida Jacqueline Toribio. 2017. Metrics for modeling code-switching across corpora. In *Proc. of the Interspeech Special Session on Code Switching*.

Yamuna Kachru. 2006. Mixers lyricing in hinglish: Blending and fusion in indian pop culture. *World Englishes* 25(2):223–233.

Eva Lösch. 2007. The construction of social distance through code-switching: an exemplary analysis for popular indian cinema. *Department of Linguistics, Technical University of Chemnitz* .

Yael Maschler. 1991. The language games bilinguals play: language alternation at language boundaries. *Language and communication* 11(2):263–289.

Yael Maschler. 1994. Appreciation ha'araxa 'o ha'arasta? [valuing or admiration]. *Negotiating contrast in bilingual disagreement talk* 14(2):207–238.

Abu Melhim and Abdel Rahman. 1991. Code-switching and linguistic accommodation in arabic. In *Perspectives on Arabic Linguistics III: Papers from the Third Annual Symposium on Arabic Linguistics*. John Benjamins Publishing, volume 80, pages 231–250.

Carol Myers-Scotton. 2005. *Multiple voices: An introduction to bilingualism*. Wiley-Blackwell.

Dong Nguyen, A Seza Doğruöz, Carolyn P Rosé, and Franciska de Jong. 2016. Computational sociolinguistics: A survey. *Computational Linguistics* .

Miwa Nishimura. 1995. A functional analysis of Japanese/English code-switching. *Journal of Pragmatics* 23(2):157–181.

Rana D. Parshad, Suman Bhowmick, Vineeta Chand, Nitu Kumari, and Neha Sinha. 2016. What is India speaking? Exploring the "Hinglish" invasion. *Physica A* 449:375–389.

Nanyun Peng, Yiming Wang, and Mark Dredze. 2014. Learning polylingual topic models from code-switched social media documents. In *ACL (2)*. pages 674–679.

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Sekhar Maddila. 2017. Estimating code-switching on twitter with a novel generalized word-level language detection technique. In *Proc of ACL 2017*.

Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury, and Niloy Ganguly. 2016. Understanding language preference for expression of opinion and sentiment: What do Hindi-English speakers do on Twitter? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Rosaura Sanchez. 1983. *Chicano discourse*. Rowley, Newbury House.

Royal Sequiera, Monojit Choudhury, Parth Gupta, Paolo Rosso, Shubham Kumar, Somnath Banerjee, Sudip Kumar Naskar, Sivaji Bandyopadhyay, Gokul Chittaranjan, Amitava Das, and Kunal Chakma. 2015. Overview of fire-2015 shared task on mixed script information retrieval. In *Working Notes of FIRE*.

A. Sharma, S. Gupta, R. Motlani, P. Bansal, M. Srivastava, R. Mamidi, and D.M Sharma. 2016. Shallow parsing pipeline for hindi-english code-mixed social media text. In *Proceedings of NAACL-HLT*.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for english-spanish code-switched text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1051–1060.

Thamar Solorio and Yang Liu. 2010. Learning to Predict Code-Switching Points. In *Proc. EMNLP*.

VINITI Vaish. 2011. Terrorism, nationalism and westernization: Code switching and identity in bollywood. *FM Hult, & KA King, K. A (Eds.). Educational linguistics in practice: Applying the local globally and the global locally* pages 27–40.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. POS Tagging of English-Hindi Code-Mixed Social Media Content. In *Proc. EMNLP*. pages 974–979.

Li Wei. 2002. what do you want me to say?on the conversation analysis approach to bilingual interaction. *Language in Society* 31(2):159–180.

Towards Normalising Konkani-English Code-Mixed Social Media Text

Akshata Phadte
DCST Goa University
Goa, India.
akshataph07@gmail.com

Gaurish Thakkar
DCST Goa University
Goa, India.
thak123@gmail.com

Abstract

In this paper, we present an empirical study on problem of word-level language identification and text normalization for Konkani-English Code-Mixed Social Media Text (CMST). we describe a new dataset which contains of more than thousands posts from Facebook posts that exhibit code mixing between Konkani-English. To the best of our knowledge, our work is the first attempt at the creation of a linguistic resource for this language pair which will be made public and developed a language identification and Normalisation System for Konkani-Englsih language pair.

We also present word-level language identification experiments are performed using this dataset. Different techniques are employed, including a simple unsupervised dictionary-based approach, supervised word-level Language identification using sequence labelling using Conditional Random Fields based models, SVM, Random Forest. The targeted research problem also entails solving another problem, that to correct English spelling errors in code-mixed social media text that contains English words as well as Romanized transliteration of words from another language, in this case Konkani.

1 Introduction

Social media in todays world possess enormous amount of data. But the problem starts in Multilingual speakers tend to exhibit code-mixing and code-switching in their use of language on social media platforms. Now Automatic understanding of Social Media text is unravelling a whole new field of study. English is still found the most popular language in Social Media Text, its dominance is receding. Code mixing occurs due to various reasons. According to a work by (Hidayat, 2012), There are the following major reasons for Code-Mixing:-

- **45%: Real lexical needs :** For instance someone is thinking of some object but is not able to recall the word in the language, then he/she will tend to switch to a language where he knows the appropriate word.
- **40%: Talking about a particular topic people :** tend to talk about some topics in their mother tongue (like food) and generally while discussing science people tend to switch to English.
- **5%: for content clarification :** while explaining one topic, for better clarification of the audience, to make the audience more clear about the topic, code switching is used.

Konkani-English bilingual speakers produce huge amounts of code-mixed social media text (CSMT). (Vyas et al., 2014) noted that the complexity in analyzing code-mixed social media text (CSMT) stems from nonadherence to a formal grammar, spelling variations, lack of annotated data, inherent conversational nature of the text and ofcourse, code-mixing. Therefore, there is a need to create datasets and Natural Language Processing (NLP) tools for code-mixed social media text (CSMT) as traditional tools are ill-equipped for it. Taking a step in this direction, we describe the Word Level Language Identification system for Konkani-English language pair that we will be building in this study. The salient contributions of this work are in formalizing the problem and related challenges for processing of Konkani-English social media data, creation of an annotated

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 85–94

dataset and initial experiments for language identification of this data.

2 Related Work

A lot of work has been done on social media data and code-mixed data over the past decades. Code-mixing being a relatively newer phenomena has gained attention of researchers only in the past two decades. On the other hand, Language Identification has been considered to be a solved problem by (McNamee, 2005), but new complications were added to this task in the context of code-mixed social media data. Similarly, Word Normalisation has been extensively studied, but there is little work done on the Konkani-English , Hindi-English language pair.

2.1 Code-Mixed Data

One of the earliest works on code-Mixing for Facebook data was done by (Hidayat, 2012) and showed that Facebook users tend to mainly use inter-sentential switching over intra-sentential, and report that 45% of the switching was instigated by real lexical needs, 40% was used for talking about a particular topic, and 5% for content clarification .

(Dey and Fung, 2014) also investigated the rules for code-switching in Hindi-English data by interviewing bilingual students and transcribing their utterances . They found that on average, roughly 67% of each sentence were made up of Hindi words and 33% English words.

2.2 Language Identification

Previous work on text has mainly been on identifying a language from documents of several languages, such that even when evidence is collected at word level, evaluation is at document level (Prager, 1999); (Singh and Gorla, 2007); (Yamaguchi and Tanaka-Ishii, 2012). (Carter et al., 2013) collected tweets in five different European languages and analysed multi-lingual microblogs for understanding the dominant language in any specific tweet . He then performed post-level language identification, experimenting with a range of different models and a character n-gram distance metric, reporting a best overall classification accuracy of 92.4%. (Tratz et al., 2013) on the other hand worked on highly code mixed tweets, with 20.2% of their test and development sets consisting of tweets in more than one language. They aimed to separate Romanised Moroccan, Arabic (Darija), English and French tweets using a Maximum Entropy classifier, achieving F-scores of 0.928 and 0.892 for English and French, but only 0.846 for Darija due to low precision.

(Nguyen and Dogruoz, 2013) worked on language identification at the word level on randomly sampled Turkish-Dutch posts from an online chat forum . They compared dictionary based methods to statistical ones. Their best system reached an accuracy of 97.6%, but with a substantially lower accuracy on post level (89.5%), even though 83% of the posts actually were monolingual. They report on language identification experiments performed on Turkish and Dutch forum data. Experiments have been carried out using language models, dictionaries, logistic regression classification and Conditional Random Fields. They find that language models are more robust than dictionaries and that contextual information is helpful for the task.

Furthermore, (Barman et al., 2014) investigated language identification at word level on Bengali-Hindi-English code-mixed social media text . They annotated a corpus with more than 180,000 tokens and achieved an accuracy of 95.76% using statistical models with monolingual dictionaries.

3 Normalisation

Owing to massive growth of SMS and social media content, text normalisation systems have gained attention where the focus is on conversion of these tokens into standard dictionary words.The first Chinese monolingual chat corpus was released by (Wong and Xia, 2008). They also introduced a word normalisation model, which was a hybrid of the Source Channel Model and phonetic mapping model.

(Wang et al., 2009) work with abbreviations for spoken Chinese rather than for English text messages. They first perform an abbreviation generation task for words and then reverse the mapping in a look-up table. They use conditional random fields as a binary classifier to determine the probability of removing a Chinese character to form an abbreviation. They rerank the resulting abbreviations by using a length prior modeled from their training data and co-occurrence of the original word and generated abbreviation using web search.

A commonly accepted research methodology is

treating normalisation as a noisy channel problem. (Choudhury et al., 2010) explain a supervised noisy channel framework using HMMs for SMS normalisation. This work was then extended by (Cook and Stevenson, 2009) to create an unsupervised noisy channel approach using probabilistic models for common abbreviation types and choosing the English word with the highest probability after combining the models. (Beaufort et al., 2010) combine a noisy channel model with a rule-based finite-state transducer and got reasonable results on French SMS, but did not test their method on English text. (Xue et al., 2011) adopted the noisy-channel framework for normalisation of microtext and proved that it is an effective method for performing normalisation.

(Vyas et al., 2014) worked on POS tagging for Hindi-English data . For Hindi normalisation, they used the system built by (Gella et al., 2013) but they did not normalise English text as they used the (Owoputi et al., 2013) Twitter POS Tagger in the next step, which does not require normalised data.

4 Data Preparation

For performing Language identification for Konkani-English language we don`t have sufficient annotated datasets and other resources. As a part of this research work we developed the following resources.

we collected data from Facebook public pages of Konkani group. All these pages are very popular with 9800 likes. A total of 4983 posts were scrapped from Konkani group pages, which were published between 6 may 2014 to 28th September 2016 and preference was given to posts having a long thread of posts. The corpus thus generated has 4,983 posts and 1,13,578 words. Due to the usage of Facebook as the underlying crowd sourcing engine, the data generated was highly conversational and had reasonable amount of social-media lingo.

Facebook posts were broken down into sentences using sentence Tokenize and 5088 of those code-mixed sentences were randomly selected for manual annotation. The data was semi-automatically cleaned and formatted, removing user names for privacy. The names of public figures in the posts were retained.

4.1 Data Statistics

The size of the original data was 34036 sentences of facebook post. 5088 (14.94%) of those code-mixed sentences were randomly selected, containing a total of 60,118 tokens. Table 1 show the distribution of the dataset at token level respectively. Of these tokens, 34,118 (56.75%) are Konkani words which are in Roman script, 17,764 (29.54%) are English words. 8,236 (13.69%) are acronym, slag words, hindi words etc which are marked as ‘ Rest ’ .

Language	All Sentences
Konkani	34,118 (56.75%)
English	17,764 (29.54%)
Rest	8236 (13.69%)
Total	**60,118**

Table 1: Data distribution at token level

4.2 Dataset examples

1. ***Interviewer:*** *Tuka British Accent’n ulopak kalta? thn plz speeak.. pleeeaase! thn i cn say ur genuis*
   Interviewer: *Bare ulon dakhoi*

The dataset is comprised of sentences similar to *Example 1* . *Example 1* shows Code-Mixing as some English words are embedded in a Konkani utterance. Spelling variations (ur - your), ambiguous words (To - So in Konkani or To in English) and non-adherence to a formal grammar (out of place ellipsis.., no or misplaced punctuation) are some of the challenges evident in analyzing the examples above.

5 Annotation Guidelines

The creation of this linguistic resource involved Language identification layer. In the following paragraphs, we describe the annotation guidelines for these tasks in detail. Manual Annotation was done on the following layer:

5.1 Language Identification

Similar to (Barman et al., 2014), we will be treating language identification as a three class (‘ kn’, ‘ en’, ‘ rest’) classification problem. Every word was given a tag out of three - en, kn and rest to mark its language. Words that a bilingual speaker could identify as belonging to either Konkani or English were marked as ‘kn’ or ‘en’, respectively.

The label ‘rest’ was given to symbols, emoticons, punctuation, named entities, acronyms, foreign words.

The label ‘rest’ was created in order to accommodate words that did not strictly belong to any language, described below:

1 Symbols, emoticons and punctuation

2 **Named Entities** : Named Entities are language independent in most cases. For instance, ‘Jack’ would be represented by equivalent characters in Konkani and English.

3 **Acronyms**: This includes SMS acronyms such as ‘LOL’, and established contractions such as ‘USA’. Acronyms are very interesting linguistic units, and play an important role in social media text. They represent not just entities but also phrases and reactions. We wanted to keep their analysis separate from the rest of the language; and hence they were categorised as ‘rest’ in our dataset.

4 **foreign words** : A word borrowed from a language except Konkani and English has been treated as ‘rest’ as well. This does not include commonly borrowed Hindi words in Konkani; they are treated as a part of Konkani language.

5 **Sub-lexical code-mixing** : Any word with word-level code-mixing has been classified as ‘rest’, since it represents a more complex morphology.

5.2 Normalisation

Words with language tag ‘ kn’ in Roman script were labeled with their standard form in the native script of Konkani Devanagari, i.e. a back-transliteration will be perform. Words with language tag ‘ en’ were labeled with their standard spelling. Words with language tag ‘ rest’ were kept as they are.
Following are some case-specific guidelines.

1 In case a token consists of two words (due to an error in typing the space), the tokens are separated and written in their original script. For instance, ‘ whatis’ would be normalised to ‘ what is’, with the language ID as English.

2 In cases where multiple spellings of a word are considered acceptable, we have allowed both spelling variations to exist as the standard spellings. For instance, in ‘ color’ and ‘ colour’, ‘ dialogue’ and ‘ dialog’, both spellings are valid.

3 Contractions such as ‘ don’t’ and ‘ who’s’ have been left undisturbed. The dataset thus contains both variations - ‘ don’t’ and ‘ do not’, depending on the original chat text.

4 Konkani has evolved through the past decades, and often we see variations in spelling of a single word. We observed the variation patterns and choose the standard spellings.

The overall annotation process was not a very ambiguous task and annotation instruction was straight-forward. Three Konkani-English bilingual speaker annotated whole dataset. They were not Linguist! Two other annotators reviewed and cleaned it. To measure inter-annotator agreement, another annotator read the guidelines and annotated 125 sentences from scratch. The inter-annotator agreement calculated by third annotator using Cohens Kappa (Cohen, 1960) came out to be 0.78 for language identification.

6 Tools and Resources

We have used the following resources and tools in our experiment. Our English dictionaries Statistics are those described in Table 2 (BNC[1] , LexNorm-List [2]) and the training set words.

Resources are :-

1. **British National Corpus (BNC):** We compile a word frequency list from the BNC (Aston and Burnard, 1998).

2. **Lexical Normalization List (LexNorm-List):** Lexical normalization dataset released by (Han and Baldwin, 2011) which consists of 41118 pair of unnormalized and normalized words / phrases.

3. **slang words:** Dictionary of Internet slang words was extracted from `http://www.noslang.com`.

4. **Transliteration pairs:** We developed wordlists for English - Konkani language

[1] `http://www.natcorp.ox.ac.uk/`

[2] We use a lexical normalization dictionary created by Han et al. (2012)

pairs using ILCI[3]. The wordlists contained few overlapping words.

source Language	Words
BNC	7,60,089
LEXNORM	41,118
Konkani Dictionary[4]	15,195

Table 2: Statistics of English and Konkani Dictionary

7 Experiments and Results

7.1 Language Identification

While language identification at the document level is a well-established task (Myers-Scotton, 1982), identifying language in social media posts has certain challenges associated to it. Spelling errors, phonetic typing, use of transliterated alphabets and abbreviations combined with code-mixing make this problem interesting. Similar to (Barman et al., 2014), we performed experiments treating language identification as a three class (‘ kn’, ‘ en’, ‘ rest’) classification problem.

For the initial experimentation, the tokenized corpus of 5088 sentences is randomly shuffled and the first 80% of dataset included in the training and the remaining 20% for testing. Since our training data is entirely labelled at the word-level by human annotators, we address the word-level language identification task in a fully supervised way. Manual annotation is a laborious process.

We address the problem of Language Identification in two different ways:

1. A simple heuristic-based approach which uses a combination of our dictionaries to classify the language of a word.

2. Word-level Language Identification using supervised machine learning with SVMs[6], Random forest and sequence labelling using CRFs[7], employing contextual information.

7.1.1 Dictionary-Based Detection

A simple rule-based method is applied to predict language of a word <w1 w2 w3 w4....wn>.
A token is considered as (‘en’,‘kn’,‘rest’) class to mark its language. if any of the following conditions satisfies.

Steps are as follows.

1. Tokenise given input query.

2. Match the word in English dictionary. so; **Wen** <w1 w2 w4 ....wn >Set of words which are found in English dictionary, found words were tags as en (English word).

3. Remaining words were compared with Konkani Dictionary[8] which is described in sections 6, **Wkn** <w2 w3 w6.... wn >, found words were tags as kn (Konkani word).

4. Set of Words **Wrest** <w5 >which remains untag are tag as ‘rest’.

5. take **Wen** set and compared with Konkani Dictionary .

6. if we found any word from **Wen** set in Konkani Dictionary than we remove that word from **Wen** set and tag the word as ‘rest’. so, we get ambiguous words. Other words remaining in set Wen are tagged as en (English words). By this approach we get particular Konkani words and English words and ambiguous words.

Dictionary	Accuracy(%)
BNC + Konkani Dictionary	69.05
LexNorm + Konkani Dictionary	68.76
BNC + LexNorm + Konkani Dictionary	69.85

Table 3: Results of dictionary-based detection

Table 3 shows the results of dictionary-based detection. We try different combinations with the above dictionaries (described in section 5). We find that using a normalized frequency is helpful and that a combination of LexNormList and Konkani-English Transliteration pairs, BNC is suited best for our data. Hence, we consider this as our baseline language identification system

7.1.2 Word-Language Detection using machine learning classifier

Word level language detection from code-mixed text can be defined as a classification problem. SVMs were chosen for the experiment (Joachims,

[3]Indian Language Corpora Initiative corpus
[6]http://scikit-learn.org/stable/
[7]https://taku910.github.io/crfpp/
[8](Konkani-English Transliteration pairs) which is described in section 5.

1998). The reason for choosing SVMs is that it currently is the best performing machine learning technique across multiple domains and for many tasks, including language identification (Baldwin and Lui, 2010). Another possibility would be to treat language detection as sequence labelling tasks (Lafferty et al., 2001); previous work (King and Abney, 2013) has shown that it provides good performance for the language identification task as well. The features used can be broadly grouped as described below:

1. **Capitalization Features**: They capture if letter(s) in a token has been capitalized or not. The reason for using this feature is that in several languages, capital Roman letters are used to denote proper nouns which could correspond to named entities. This feature is meaningful only for languages which make case distinction (e.g., Roman, Greek and Cyrillic scripts).

2. **Contextual Features**: They constitute the current and surrounding tokens and the length of the current token. Code-switching points are context sensitive and depend on various structural restrictions.

3. **Special Character Features**: They capture the existence of special characters and numbers in the token. Tweets contain various entities like hashtags, mentions, links, smileys, etc., which are signaled by #, and other special characters.

4. **Lexicon Features**: These features indicate the existence of a token in lexicons. Common words in a language and named entities can be curated into finite, manageable lexicons and were therefore used for cases where such data was available.

5. **Character n-gram features**: we also used charagter n-grams for n=1 to 5.

We perform experiments with an different classifier for different combination of these features. The features are listed in Table 4. The accuracies with respect to different classifier and Features are shown in Table 5. All possible combinations are considered during experiments. It can be seen from the results that character gram feature provides best results . Whereas for lexical and Word gram and Contextual features provides comparable results.

7.1.2.1 System Accuracy

The approach using CRFs had a greater accuracy, which validated our hypothesis and also proved that context is crucial in this process. The results of this module are shown in Table 5.

7.2 Normalisation

Once the language identification task is complete, there will be a need to convert the noisy non-standard tokens (such as English and Konkani words inconsistently written in many ways using the Roman script) in the text into standard words. To fix this, a normalization module that performs language-specific transformations, yielding the correct spelling for a given word was built. we had used two approach for normalisation.

1)Konkani Transliterator and Normalizer
2) Noisy Channel Framework.

These are further explained in Section 7.2.1 and 7.2.2

7.2.1 Konkani Transliterator and Normalizer (Normalizer):

We use CMU Part of Speech tagger [9] on English words which reported an accuracy of 65.39% , it normalizes English words as a primary step. We used Python-Irtrans [10] developed by IIIT-Hyderabad for transliteration of Konkani words from Roman to Devanagari. We ran the konkani words on transliteration system in order to normalize it. This tool is used to convert roman into Konkani script i.e Python-Irtrans which reported an accuracy of 60.09%.

7.2.2 Noisy Channel Framework:

For transliterating the detected Romanized Konkani words and for noisy English words, we built A Two Layer Normalizer was built for both Konkani and English.

1. **Compression**

2. **Normalizer**

The message is processed using the following techniques described in following sections.

1. Compression: In Social Media platform, while chatting, users most of the time express their emotions/mood by stressing over a few characters

[9] http://www.cs.cmu.edu/ark/
[10] https://github.com/irshadbhat/indic-trans

ID	Feature Description	Type
Capitalization Features		
CAP1	Is first letter capitalized?	True/False
CAP2	Is any character capitalized?	True/False
CAP3	Are all characters capitalized?	True/False
Contextual Features		
CON1	Current Token	String
CON2	Previous 3 and next 3 tokens	String
CON3	Word length	String
Special Character Features		
CHR0	Is English alphabet word?	True/False
CHR1	Contains @ in locations 2-end	True/False
CHR2	Contains # in locations 2-end	True/False
CHR3	Contains ' in locations 2-end	True/False
CHR4	Contains / in locations 2-end	True/False
CHR5	Contains number in locations 2-end	True/False
CHR6	Contains punctuation in locations 2-end	True/False
CHR7	Starts with @	True/False
CHR8	Starts with #	True/False
CHR9	Starts with '	True/False
CHR10	Starts with /	True/False
CHR11	Starts with number	True/False
CHR12	Starts with punctuation	True/False
CHR13	Token is a number?	True/False
CHR14	Token is a punctuation?	True/False
CHR15	Token contains a number?	True/False
Lexical Features		
LEX1	Is present in English dictionary?	True/False
LEX2	Is Acronym	True/False
LEX3	Is NE?	True/False
Character n-gram Features		
CNG0	Uni-gram, bigram,trigram	vector
Word n-gram Feature		
WNG0	Uni-gram, bigram,trigram	Probability

Table 4: A description of features used.

in the word. *For example*, usage of words are *thanksss, sryy, byeeee, wowww, gooooood* which corresponds the person being obliged, needy, apologetic, emotional, amazed, etc.

As we know, it is unlikely for an English word to contain the same character consecutively for three or more times hence, we compress all the repeated windows of character length greater than two, to two characters.

Each window now contains two characters of the same alphabet in cases of repetition. Let n be the number of windows, obtained from the previous step. Since average length of English word (Mayzner and Tresselt, 1965) is approximately 4.9, we apply brute force search over 2n possibilities to select a valid dictionary word. If none of the combinations form a valid English word, the compressed form is used for normalization.

Table 6 contains sanitized sample output from our compression module for further processing.

2. Normalizer: Text Message Normalization is the process of translating ad-hoc abbreviations, typographical errors, phonetic substitution and ungrammatical structures used in text messaging (SMS and Chatting) to plain English. Use of such language (often referred as Chatting Language)

Features	System		
	SVM	RF	CRF
CON*	0.86	0.89	0.89
CHR*	0.87	0.86	0.87
CAP*	0.887	0.88	0.877
LEX*	0.89	0.89	0.88
CNG*	0.93	0.92	0.94
WNG*	0.89	0.88	0.89
ALL	0.898	0.90	0.97

Table 5: System word-level accuracies (in %) for language detection from code-mixed text on the test datasets.'*" is used to indicate a group of features. Refer Table. 4 for the feature Ids.

Input Sentence	Output Sentence
I am so gooood	I am so good !
tuuu kaashe asa...	tu kashe asa...

Table 6: Sample output of Compression module

induces noise which poses additional processing challenges. While dictionary lookup based methods [11] are popular for Normalization, they can not make use of context and domain knowledge. **For example**, *yr* can have multiple translations like *year, your*.

We tackle this by building our normalzation system based on the state-of-the-art Phrase Based Machine Translation System (PB-SMT), that learns normalization patterns from a large number of training examples. We use Moses (Koehn et al., 2007), a statistical machine translation system that allows training of translation models.

PB-SMT is a machine translation model; therefore, we adapted the PB-SMT model to the transliteration task by translating characters rather than words as in character-level translation. For character alignment, we used GIZA++ implementation of the IBM word alignment model. To suit the PB-SMT model to the transliteration task, we do not use the phrase reordering model. The target language model is built on the target side of the parallel data with Kneser-Ney (Kneser and Ney, 1995) smoothing using the IRSTLM tool (Federico et al., 2008). In a bid to simulate syllable level transliteration we also built a Normalization model by breaking the English and Konkani words to chunks of consecutive characters and trained the transliteration system on this chunked data.

Training process requires a Language Model of the target language and a parallel corpora containing aligned un-normalized and normalized word pairs.

For English and Konkani word Normalization, our language model consists of 50,156 English un-normalized and normalized words taken from the web, 15195 Konkani words taken from Indian Language Corpora Initiative (ILCI) Corpus and manually transliterated.

Parallel corpora was used which is described in section 6.

Table 7. presents the obtained results.

7.2.2.1 System Accuracy

The accuracy of this system is shown in Table 7. The accuracy for the Konkani normaliser is higher than that for English.

Languages	Accuracy (%)
English Normalizer	72.81
Konkani Normalizer	77.21

Table 7: Token level Normalization Accuracy

8 Conclusion and Future Work

We have presented an initial study on automatic language identification and text normalisation with Indian language code mixing from social media communication. This is a quite complex language identification task which has to be carried out at the word level, since each message and each single sentence can contain text and words in several languages. The paper has aimed to put the spotlight on the issues that make code-mixed text challenging for language processing. we have focused on the process of creating and annotating a much needed dataset for code-mixed Konkani-English sentences in the social media context, as well as developed language identification and normalisation systems follow supervised machine learning and report final accuracies of 97.01% and 72.81% for English Normalizer , 77.81% for Konkani Normalizer for our dataset, respectively.

In the future, we intend to continue creating more annotated code-mixed social media data. We intend to use this dataset to build tools for code-mixed data like POS taggers, morph analysers, chunkers and parsers. In the future we would also like to evaluate on adding more language classes,

[11] http://www.lingo2word.com

particularly for named entities and acronyms influences the overall accuracy of our system.

References

Guy Aston and Lou Burnard. 1998. *The BNC handbook: exploring the British National Corpus with SARA*. Capstone.

Timothy Baldwin and Marco Lui. 2010. Language identification: The long and the short of the matter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 229–237.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. *EMNLP 2014* 13.

Richard Beaufort, Sophie Roekhaut, Louise-Amélie Cougnon, and Cédrick Fairon. 2010. A hybrid rule/model-based finite-state framework for normalizing sms messages. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 770–779.

Simon Carter, Wouter Weerkamp, and Manos Tsagkias. 2013. Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text. *Language Resources and Evaluation* 47(1):195–215.

Monojit Choudhury, Kalika Bali, Tirthankar Dasgupta, and Anupam Basu. 2010. Resource creation for training and testing of transliteration systems for indian languages. LREC.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20(1):37–46.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization. In *Proceedings of the workshop on computational approaches to linguistic creativity*. Association for Computational Linguistics, pages 71–78.

Anik Dey and Pascale Fung. 2014. A hindi-english code-switching corpus. In *LREC*. pages 2410–2413.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. Irstlm: an open source toolkit for handling large scale language models. In *Interspeech*. pages 1618–1621.

Spandana Gella, Jatin Sharma, and Kalika Bali. 2013. Query word labeling and back transliteration for indian languages: Shared task system description. *FIRE Working Notes* 3.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 368–378.

Taofik Hidayat. 2012. An analysis of code switching used by facebookers (a case study in a social network site). In *Student essay for the study programme-PendidikanBahasaInggris (English Education) at STKIP Siliwangi Bandung*.

Thorsten Joachims. 1998. Making large-scale svm learning practical. Technical report, Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund.

Ben King and Steven P Abney. 2013. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *HLT-NAACL*. pages 1110–1119.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. IEEE, volume 1, pages 181–184.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, pages 177–180.

John Lafferty, Andrew McCallum, Fernando Pereira, et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*. volume 1, pages 282–289.

Mark S Mayzner and Margaret Elizabeth Tresselt. 1965. Tables of single-letter and digram frequency counts for various word-length and letter-position combinations. *Psychonomic monograph supplements* .

Paul McNamee. 2005. Language identification: a solved problem suitable for undergraduate instruction. *Journal of Computing Sciences in Colleges* 20(3):94–101.

Carol Myers-Scotton. 1982. Duelling languages: Grammatical structure in codeswitching. In *Oxford University Press.*.

Dong-Phuong Nguyen and A Seza Dogruoz. 2013. Word level language identification in online multilingual communication. Association for Computational Linguistics.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

John M Prager. 1999. Linguini: Language identification for multilingual documents. *Journal of Management Information Systems* 16(3):71–101.

Anil Kumar Singh and Jagadeesh Gorla. 2007. Identification of languages and encodings in a multilingual document. In *Building and Exploring Web Corpora (WAC3-2007): Proceedings of the 3rd Web as Corpus Workshop, Incorporating Cleaneval*. Presses univ. de Louvain, volume 4, page 95.

Stephen Tratz, Douglas Briesch, Jamal Laoudi, and Clare Voss. 2013. Tweet conversation annotation tool with a focus on an arabic dialect, moroccan darija. *LAW VII & ID* 135.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. In *EMNLP*. volume 14, pages 974–979.

Min Wang, Chen Yang, and Chenxi Cheng. 2009. The contributions of phonology, orthography, and morphology in chinese–english biliteracy acquisition. *Applied Psycholinguistics* 30(02):291–314.

Kam-Fai Wong and Yunqing Xia. 2008. Normalization of chinese chat language. *Language Resources and Evaluation* 42(2):219–242.

Zhenzhen Xue, Dawei Yin, and Brian D Davison. 2011. Normalizing microtext. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Hiroshi Yamaguchi and Kumiko Tanaka-Ishii. 2012. Text segmentation by language using minimum description length. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 969–978.

Towards developing a phonetically balanced code-mixed speech corpus for Hindi-English ASR

Ayushi Pandey
IIIT-Hyderabad
Hyderabad
email@domain

Brij Mohan Lal Srivastava
Microsoft Research
Bangalore, India
t-brsriv@microsoft.com

S V Gangashetty
IIIT-Hyderabad
Hyderabad
svg@iiit.ac.in

Abstract

This paper presents the ongoing process in the design of the first phase of the phonetically balanced code-mixed corpus of Hindi-English speech (PBCM-Phase I). The reference corpus is a large code-mixed (LCM) newspaper corpus selected from the sections that contain frequent English insertions in a matrix of Hindi sentence. From a phonetically transcribed corpus, compulsory inclusion of lowest frequency triphones has been ensured, with the assumption that high frequency phones may automatically be included. A high correlation of 0.81 with the representative large corpus has been observed. A small scale speech corpus of 5.6 hours has been collected, by the contribution of 4 volunteer native Hindi speakers. The recording has been conducted in a professional recording studio environment. As a second contribution, this paper also presents a baseline recognition system with pooled monolingual and code-mixed speech datasets as training and testing environments.

1 Introduction

Code-mixing is a frequently encountered phenomenon in day-to-day communication in multilingual and bilingual communities. The phenomenon is so widespread that is often considered a different, emerging variety of the language. In India, English has been granted the status of an official language by the constitution. Additionally, there are complex diglossic patterns existing between most of the regional languages and English, where English is usually the language of prestige. Indian bilingual speakers therefore, show abundant mixing and switching between their regional language and English. Computational modeling of the phenomenon of code-mixing and code-switching assumes particular relevance with the advancement of social media. However, computational studies for both textual and speech processing of code-mixing suffer from a sincere disadvantage: lack of data.

To investigate the problem in a controlled environment, the paper presents the first phase of a Phonetically Balanced Code-Mixed (PBCM-Phase I) read speech corpus.

The design of the paper is as follows: Section 2 elaborates the popular methods in the area of corpus design. Section 3 details the metric in use for designing the speech corpus. Section 4 details the recording procedure and information about speakers. Section 5 provides a brief introduction to DNN based acoustic modeling, language modeling and an adaptive implementation of both in bilingual speech recognition. Section 6 presents the results and concludes the paper.

2 Related studies in corpus develo

It is commonly believed that the quality of the training data for nearly all speech processing systems, largely determines the success of the systems. Adequate phonemic coverage with minimal redundancy is crucial in corpus design, to allow for a wide coverage of common phonetic forms in a variety of their contexts. A large and usually diverse text corpus serves as a **reference** corpus, from which a set of phonetically rich and/or balanced sentences are selected. Phonetically rich sentences (Radová and Vopálka, 1999) contain a homogeneous frequency distribution of

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 95–101

Table 1: Genre-wise distribution: LCM corpus

Section	Number of sentences
Lifestyle	9,495
Sports	11,202
Gadgets and Technology	11,342

all phonemes in the language. In a phonetically rich corpus, adequate training instances of almost all phones, or at least one instance of every phone are compulsorily included. In a phonetically balanced corpus, the distribution of phones is modeled to be proportionate to the natural phonemic distribution in the concerned language. Once the phonetic transcriptions are made available along with the speech recordings, the *add-on* procedure is a popular method (Falaschi, 1989). From the reference corpus, a set of sentences are randomly selected as the seed corpus. Thereafter, sentences with frequency scores proportionate to those of the already selected corpus are chosen. Speech databases designed especially for recognition studies benefit from a context-sensitive phone; for example a triphone or another subword unit like a syllable or a diphone. Santen et al (Van Santen and Buchsbaum, 1997) note that a training corpus requires to be prepared towards less frequent phonetic units. To optimize coverage of all phonetic units, ASR studies usually implement a sentence selection approach with weighted frequencies of triphones, where the weights are actually the inverse of frequencies. This ensures an inclusion of rare phones in the corpus, while the high frequency phones are collected inadvertently. (Van Santen and Buchsbaum, 1997). In India, there has been heavy investment on developing corpora that are both phonetically rich and/or balanced. But most of these have been designed for monolingual speech recognition purposes, and do not cover the scope of code-mixing. The next section details our approaches in design and development of PBCM-Phase I, the first phase of a phonetically balanced code-mixed speech corpus for an Indian language pair.

3 Design of the data corpus

The nature of code-mixing has best been seen reflected in conversational communication, because the practice of code-mixing is still frowned upon in formal registers. However, owing to an increasing readership, selected sections (like Sports, Technology, Lifestyle) of newspapers offer enormous coverage of code-mixing. In addition to a wide and diverse coverage, these sections have also introduced a standardization into code-mixed diction. In this paper, we design a representative corpus, the Large Code-Mixed (LCM) Corpus as a large and diverse textual database, scraped from three sections, namely Gadgets and Technology, Lifestyle and Sports from the popular Hindi newspaper DainikBhaskar (http://epaper.bhaskar.com/). Details of these sections are given in Table 1. Figure 1 displays the code-mixing distributions in the respective genres.

Upon preliminary observation, we note that while the Sports and the Gadgets and the Technology sections have prominent technical vocabulary borrowings, this content is not always limited to lack of parallel vocabulary in the matrix language.

Example:
पाट्र्स खरीद कर टेक्नीशियन से बदलवा सकते हैं ।

Translation:
"One could buy parts and get them replaced by a technician."

3.1 Selecting sentences based on triphone frequency

In development of most ASR corpora, frequency of the triphone has been given specific importance. This is primarily because of the ability of the triphone to be sensitive to both the preceding and the succeeding context. To obtain an optimal selection of sentences, the corpus needed to be balanced not only in a set of unique phones, but also the contexts that

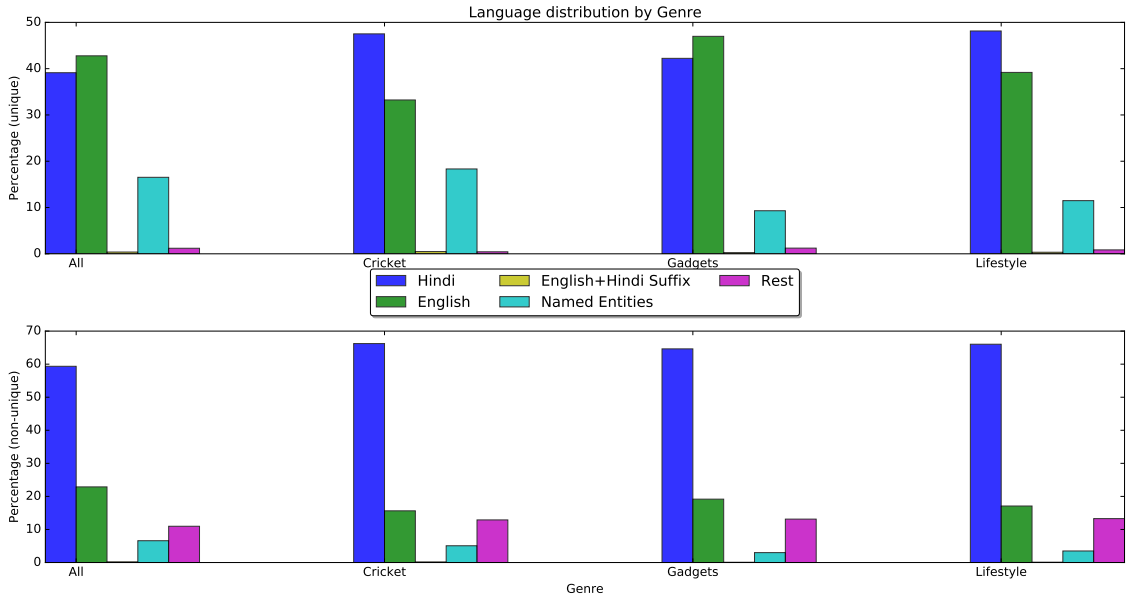


Figure 1: Stacked graphs displaying the Unique (above) vs Non-unique (below) frequency of distribution of English embeddings, Named-Entities and Rest contained in the PBCM corpus.

they occurred in.

Word-internal triphones were chosen as opposed to word-adjacent triphones, because word-internal triphone sequence can aid the identification of language at a word-level.

The design on the optimal text selection was created using the following steps:

1. As a pre-processing step for creation of a read-speech corpus, sentences only of length 5-12 words were selected.

2. Phoneme sequences for unique Roman words in the corpus were generated using a grapheme to phoneme (G2P) converter trained on 7000 words using sequence-to-sequence (Yao and Zweig, 2015) learning approach implemented in Tensorflow.

3. Phoneme sequences for uniqueDevanagari words in the corpus were converted to their corresponding WX notation representation. (Bharati et al., 1995)

4. Word-internal triphones were collected and arranged based on the descending order of their frequency of occurrence in the corpus.

5. To ensure the coverage of rare phones, all the sentences containing words that were composed of the triphones lower in frequency than the threshold, were selected.

   The Phonetically Balanced Code-Mixed (PBCM) corpus of 2,694 sentences was created based on the low frequency of the rarest triphones.

3.2 Correlation computation

After the selection process of sentences, we wanted to ensure that this is truly representative of the distribution of phones present in natural language. Unique phones from both the LCM corpus and the PBCM corpus were collected, and a Pearson's correlation was computed between them. The Pearson's correlation coefficient between the two vectors was found to be 0.81. A high correlation value display a proportionate distribution of phones between the sampled corpus PBCM, and the reference corpus LCM.

3.3 Text annotations

We are in the process of annotating this data at the following four levels: 1) language identity, 2) word-identity, 3) part-of-speech and 4) word-identity in the phonetic form. Manual annotation of language identity on the corresponding text corpus has been completed for 1,760 sentences. A percentage

Table 2: Details of speakers for the PBCM corpus

Speaker ID	Sentences	Age
FEMALE-1	675	32
FEMALE-2	673	25
MALE-1	676	28
MALE-2	671	22

distribution of English embeddings across genres is illustrated in Figure 2.
As can be clearly seen from Panel 1 of Figure 1, the Gadgets and LifeStyle section display a wider distribution of unique English word embeddings in their respective content.

4 Recording procedure

After the sentence selection procedure is completed, the next step is to conduct the actual recordings. This section presents a detailed description of volunteer speakers, recording environment and the equipment setup utilized for recording. The duration of recorded utterances collected so far is **5.6 hours**.

4.1 Speakers description

Speech recordings were collected from 4 volunteer speakers (2 male and 2 female), who were each a native speaker of Hindi and had received education in English medium schools. The age range of these speakers was between 20-35 years. Sentences from the designed PBCM corpus was equally divided among the speakers, so that every speaker recorded around 675 sentences. Exact details of speakers can be found in Table 2.

At this stage, the corpus reflects a balance in phonetic coverage, but low acoustic variability in terms of speakers. We are planning to develop this corpus into a large corpus of about 100 speakers, with a more vast collection of speech utterances.

4.2 Recording environment and equipment

The recording of the speech utterances of the PBCM corpus was conducted in a professional voice recording studio (Deepali Studio, Lucknow, Uttar Pradesh). The recordings were administered through the Nuendo speech processing software. The equipment consisted of an integrated SoundCraft Digital-Mixer, a high fidelity noise free Sennheiser microphone and two Yamaha studio speaker systems.
The volunteer speaker was instructed to maintain a distance 10-12 inches from the microphone. Each speaker conducted recordings in a set of 20 sentences, after which they were given a water-break and vocal rest of 2-5 minutes. Before each recording session, each volunteer speaker was primed by having the sentences read out aloud to them, in order to minimize hesitation while speaking. After every 100 sentences, the speaker was given a vocal rest for 10-15 minutes.

4.3 Post-processing of audio files

The files recorded through the session were then post-processed as a final step. A long sound file of 20 utterances each was manually split into a one sound file per sentence format, using Praat. Repetitions and non-verbal sounds were also manually removed, and only noise-free sentences were compiled. For preparing the data for use for speech recognition, we gave each sound file a unique ID, which contained the speaker information and the serial number of recording. A silence of 1 second was appended to each sound file, both before and after the utterance. The sound files, initially recorded at 44 kHz and 24-bit resolution, were also downsampled to 16 kHz and a 16-bit resolution.

5 Baseline Automatic Speech Recognition

Computational modeling of the phenomenon of code-mixing assumes particular relevance with the advancement of social media in multilingual and bilingual communities. In processing of code-mixed speech, several ideas have been put forward.

5.1 The acoustic modeling component

Acoustic model aims to establish statistical relationship between speech utterances and the corresponding text.

In general, let $\boldsymbol{O} = \{x_1, ..., x_T\}$ be the acoustic observations and $\boldsymbol{w} = \{w_1, ..., w_T\}$ be the corresponding word sequence. Then the DNN must learn $p(\boldsymbol{w}|\boldsymbol{O})$, which is the conditional distribution of words given acoustic

observations. DNN acts as a discriminative classifier which classifies tied-state phoneme classes (*senones*) given the acoustic observations $\boldsymbol{O}$. The acoustic model decodes speech utterance and proposes a directed acyclic graph (*lattice*) of phonemes with edges as transition probabilities. The lattice is then searched for contesting legal hypotheses. In order to correct the errors made by the DNN acoustic model, we multiply the probabilities from the existing knowledge in form of language model. This process is called lattice rescoring. By devising statistical language models which can mimic the original structure of language, we can supplement the probability of correct hypothesis and boost the accuracy of the overall system.

Multilingual and code-mixed ASR have seen two major trends. The first approach uses a language identification system implemented at the front-end, and a monolingual speech recognizer at the back end. Once the language has been identified at the word level, the segments (words) are passed as input to monolingual speech recognizers for phoneme decoding. However, such two-pass approaches return inferior results, owing to an unpredictable error-propagation from the language identifier at the front end, to the speech recognizer at back end. To circumvent this error, a one-pass approach is chosen, wherein the language identification component is completely removed. Some such efforts have been made by Bhuvanagiri (Bhuvanagiri and Kopparapu, 2010) et al, where they exploit an adaptation of the existing monolingual (English) training resources for code-mixed Hindi-English speech recognition. An approximation of the missing Hindi phonemes is achieved using either a direct mapping or a combination of existing English phones. Similar monolingual training resource extrapolation studies have been conducted by Fung et al, in experimenting with three sets of phonemic adaptation. (Yuen and Pascale, 1998). More approaches to merging phonesets can be seen in an interpolation of two monolingual speech corpora. (Chan et al.,). Model adaptation of monolingual corpora for code-mixed speech recognition has also been augmented with a model reconstruction with accented speech. (Li et al., 2011)

5.2 Language independent phones

One of the primary stages in the design of a code-mixed or multilingual speech recognition system, is the development of a combined phoneset. A combined phoneset allows for the recognition system to be prepared for all phones of the participating languages. If one of the languages is low in resources, then its phones are mapped to the closest approximations of phones in a high-resource language. A variety of phonemic adaptation methods have been explored, for example rule-based, manual (Bhuvanagiri and Kopparapu, 2010), (Yuen and Pascale, 1998) or clustering. (Li et al., 2011) For the purpose of this experiment, the speech utterances that are contained in the PBCM-Phase I are extracted from a Hindi national newspaper. The English embeddings are predominantly in Devanagari. However, the corpus does have a sizeable collection of sentences that have word-insertions in Roman script. To ensure phonetic consistency among all the transcripts, we use automatic transliteration (Bhat et al., 2015) to convert the words in Roman script into their respective Devanagari representation.

This experiment can further be refined through evaluating correspondence between the resulted phonetic transcription of the WX and the actual English phonetic transcription, and then intervening with a rule-based mapping. For this, however, an LID for Devanagari would need to be prepared, which is beyond the scope of the present study.

5.3 Feature selection and extraction

We propose the usage of the WX notation (Bharati et al., 1995) for establishing a grapheme to phoneme representation of the Devanagari. For the Roman utterances, the sequence-to-sequence converter has been implemented, and the phonetic representation belongs in the IPA.

The DNN model is trained over features obtained initially by concatenating ± 4 frames of MFCC followed by followed by Linear Discriminant Analysis (LDA). The features thus obtained have unit variance. These features are subjected to Maximum Likelihood Linear Transform (MLLT). MLLT is a feature-space transform with the objective function

which is defined as the sum of the average per-frame log-likelihood of the transformed features given the model, and the log determinant of the transform.

In the end, we apply feature-space Maximum Likelihood Linear Regression (fMLLR), which is an affine feature transform of the form $x \rightarrow Ax + b$. We finally obtain the 40-dimensional feature set used for DNN training.

5.4 Training and test corpora development

The main objective of the latter part of this study is to be able to utilize and adapt high-resource monolingual corpora for a low-resource setting such as code-mixed speech. In order to achieve such an extrapolation, an adaptation of phonemes has already been explained in the previous section. The training dataset comprises of a combination of monolingual Hindi speech and a small portion of code-mixed speech.The training dataset comprises of monolingual speech corpus containing speech recordings from 17 speakers, collected through the Hindi DD-News channel and Indic speech database and 3 code-mixed speakers, collected through the PBCM corpus. The testing dataset comprises of the 3 monolingual speakers and 1 code-mixed speaker, collected through the PBCM corpus.

5.5 The language modeling component

Language models are prevalent in ASR studies for providing word-level probability scores derived from the sequential structure of sentences. N-gram (trigram, 4-gram etc) language models are designed on the assumption that the probability of a given word $p(w_t)$ can be determined based on the context h_t that it is preceded by.

$$p(w_{1:T}) = \prod p(w_t|w_{t-1}w_{t-2}..w_{t-T}) = p(w_t|h_t) \quad (1)$$

Several ideas have been put forward in designing language models for code-mixed speech. Approaches relying on (Vu et al., 2012) acoustic modeling alone have been refined and augmented through modifications of the language model. Grammatical constraints (equivalence and government constraint) are implemented in (Li and Fung, 2012), to predict the correctness or likelihood of a switch in a sentence. Additionally, to circumvent the limitations of low-resources in data, class-based language models (Yeh et al., 2010), (Tsai et al., 2010) are used. Improved language modeling for code-mixed speech recognition have also aided in characterising some of the speaker specific patterns of code-mixing. (Vu et al., 2013)

6 Results and discussion

Acoustic models were trained according to Dan's NNET2 setup (Povey et al., 2014). The featureset implemented for training has been described in detail in section 5.3. We conducted two sets of experiments with respect to evaluating the scalability of the training corpus.

- **Expt 1:** The speech transcripts that belonged in the testing corpus were included in the language model training. This setup was designed so as to remove any instance of an out-of-vocabulary word, and evaluating the performance of an monolingual acoustic model with an adapted phoneset.
- **Expt 2:** The speech utterances that had been covered in the spoken corpus were excluded from the language model training. The design of this setup allowed us to evaluate the ASR based on a monolingual language modeling and monolingual acoustic modeling.

Expt 2 reveals that the WER obtained over the mixed (3 monolingual, 1 code-mixed) test set evolved from 72.34% with respect to monophone training to 41.63% for Dan's NNET2. Removing out-of-vocabulary words significantly reduces the WER, as the basline (monophone) results in a far lower error 46.54 %, when compared with Expt 1.

7 Conclusion

We present the initial design and the ongoing process in the development of a phonetically balanced speech corpus of Hindi-English non-conversational speech. Data has been subsetted from selected sections of a popular Hindi newspaper, DainikBhaskar, and a corpus of

LM	mono	tri1	tri2b	tri2b$_m mi$	tri2b$_m pe$	tri3b	tri3c	sgmm2	nnet
Expt 1	46.54	35.84	37.54	36.44	36.74	15.35	17.86	13.59	10.63
Expt 2	72.34	58.64	59.05	59.32	59.07	45.29	46.72	41.60	41.66

Table 3: Table with word error rate (WER) of different acoustic models implemented with the two language modeling setups

2,694 sentences have been collected. Sampling from a Large Code-Mixed (LCM) corpus into a Phonetically Balanced Code-Mixed corpus has been designed through a threshold frequency of triphones, with the assumption that high-frequency phones would be accommodated inadvertently through the process. The first phase of the PBCM corpus has successfully been recorded. We also present the development of a baseline automatic speech recognition system, modeled on adapting the available high-resource such as the monolingual Hindi speech corpora.

References

Akshar Bharati, Vineet Chaitanya, Rajeev Sangal, and KV Ramakrishnamacharyulu. 1995. *Natural language processing: a Paninian perspective*. Prentice-Hall of India New Delhi.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2015. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, FIRE '14, pages 48–53, New York, NY, USA. ACM.

K Bhuvanagiri and Sunil Kopparapu. 2010. An approach to mixed language automatic speech recognition. *Oriental COCOSDA, Kathmandu, Nepal*.

Joyce YC Chan, Houwei Cao, PC Ching, and Tan Lee. Automatic recognition of cantonese-english code-mixing speech.

Alessandro Falaschi. 1989. An automated procedure for minimum size phonetically balanced phrases selection. In *Speech Input/Output Assessment and Speech Databases*.

Ying Li and Pascale Fung. 2012. Code-switch language model with inversion constraints for mixed language speech recognition. In *COLING*, pages 1671–1680.

Ying Li, Pascale Fung, Ping Xu, and Yi Liu. 2011. Asymmetric acoustic modeling of mixed language speech. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5004–5007. IEEE.

Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. 2014. Parallel training of dnns with natural gradient and parameter averaging. *arXiv preprint arXiv:1410.7455*.

Vlasta Radová and Petr Vopálka. 1999. Methods of sentences selection for read-speech corpus design. In *International Workshop on Text, Speech and Dialogue*, pages 165–170. Springer.

Tsai-Lu Tsai, Chen-Yu Chiang, Hsiu-Min Yu, Lieh-Shih Lo, Yih-Ru Wang, and Sin-Horng Chen. 2010. A study on hakka and mixed hakka-mandarin speech recognition. In *Chinese Spoken Language Processing (ISCSLP), 2010 7th International Symposium on*, pages 199–204. IEEE.

Jan PH Van Santen and Adam L Buchsbaum. 1997. Methods for optimal text selection. In *EuroSpeech*.

Ngoc Thang Vu, Dau-Cheng Lyu, Jochen Weiner, Dominic Telaar, Tim Schlippe, Fabian Blaicher, Eng-Siong Chng, Tanja Schultz, and Haizhou Li. 2012. A first speech recognition system for mandarin-english code-switch conversational speech. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4889–4892. IEEE.

Ngoc Thang Vu, Heike Adel, and Tanja Schultz. 2013. An investigation of code-switching attitude dependent language modeling. In *International Conference on Statistical Language and Speech Processing*, pages 297–308. Springer.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196*.

Ching Feng Yeh, Chao Yu Huang, Liang Che Sun, and Lin Shan Lee. 2010. An integrated framework for transcribing mandarin-english code-mixed lectures with improved acoustic and language modeling. In *Chinese Spoken Language Processing (ISCSLP), 2010 7th International Symposium on*, pages 214–219. IEEE.

MA Chi Yuen and FUNG Pascale. 1998. Using english phoneme models for chinese speech recognition. In *International Symposium on Chinese Spoken language processing*, pages 80–82. Citeseer.

Keynote Lecture-2

Grammatical Error Correction: Past, Present, and Future

NG Hwee Tou
National University of Singapore

Grammatical error correction is the task of correcting errors in texts including spelling, grammar, collocation, and word choice errors. Software that can automatically correct errors in English texts has far-reaching impact, since it is estimated that more than one billion people worldwide are learning English as a second language and they will benefit greatly from such software. In this talk, I will give an overview of past and present research on grammatical error correction, and suggest future research directions.

Hybrid Approach for Marathi Named Entity Recognition

Nita Patil
SOCS
N. M. U. Jalgaon (MS)
India
nvpatil@nmu.ac.in

Ajay S. Patil
SOCS
N. M. U. Jalgaon (MS)
India
aspatil@nmu.ac.in

B. V. Pawar
SOCS
N. M. U. Jalgaon (MS)
India
bvpawar@nmu.ac.in

Abstract

This paper describes a named entity recognition system that combines hidden markov model, handcrafted rules, and gazetteers to recognize named entities in Marathi language. The objective of the system is to recognize twelve types of NEs from the Marathi text. Marathi is morphologically rich and inflectional language. The inflections in NEs are handled by using lemmatization. The difficulties of zero and poor probabilities caused due to the sparse data are handled using pseudo word replacement and smoothing techniques. Viterbi algorithm is used for decoding and word disambiguation. The performance of the system is improved using gazetteers and grammar rules.

Keywords: Named Entity Recognition, Marathi, HMM, Gazetteers, Rules

1 Introduction

Named Entity Recognition (NER) is information extraction task which can play significant role in many different natural language processing tasks such as information retrieval, machine translation, question answering systems etc. Predefined entities in text such as people, organizations, locations, events, expressions such as amount, percentage, numbers, date, time are named entities (NEs). Identification of NEs from unstructured text and their classification into suitable NE class is known as NER. This paper describes a hybrid model based on Hidden Markov Model (HMM), handcrafted rules and gazetteers to recognize named entities in Marathi. The difficulties of unseen probabilities are handled by pseudo word replacement and poor probabilities caused due to sparse data are handled using smoothing techniques. Viterbi alogirithm is used for decoding and word disambiguation. The performance of the system is improved using gazetteers. Linguistic rules are used to generate patterns that can recognize dates, time and numerical expressions. Following MUC specifications twelve types of NE are considered in recognition problem they are Person, Organization, Location, Miscellaneous, Amount, Number, Date, Time, Year, Month, Day and Measure. Patil (2017) reported the NER system based on trigram HMM model trained using pre-processed data for the Marathi language. The system uses Viterbi decoding to generate the optimal tag sequence for the test data. The system implemented using lemma model with trigram HMM has performed well in NE recognition, but it has further scope for improvement. Numerical NEs generally follow some fixed patterns, hence linguistic knowledge based recognition could be the better choice than probability based recognition. The study aims to improve NE recognition rate by combining effectiveness of statistical model with goodness of rule and gazetteer based technique for Marathi NER. The paper is organized in five main sections. Introduction and literature survey is discussed in first and second section. Supervised learning method for Marathi NER that uses HMM is described in third section. Fourth section briefs about rules and gazetteer based Marathi named entity recognition and the fifth section of the paper describes proposed hybrid model for development of Marathi NER system.

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 103–111

2 Related Work

Research in named entity recognition for Indian languages is initiated by (Bandyopadhyay (2007), Varma (2008), Murthy (2008), Nusrat (2008), Bhattacharya (2009)). Many researchers have proposed rule based NER systems (Krupka (1998), William (1998), Awaghad (2009), Kashif (2010), Sasidhar (2011)) that give accurate results and achieve high performance. But the downside of this approach is lack of robustness and portability. Also, high maintenance is needed. Recently NER problems are solved by most of the researchers using statistical machine learning approach which uses mathematical and statistical models to train and test the data. Reasonable performance is reported by using this approach by the researchers (Daniel (1999), John (2001), GuoDong (2002), Asif (2008)). One more thought towards solving NER problem is combining the goodness of both approaches to achieve great performance and minimize the drawback is using Hybrid approach (Raymond (2006), Branimir (2008), Alireza (2008), Sitanath (2009), Xueqing (2009). Hybrid approach combines hand crafted rules with machine learning techniques. The time-consuming work like creation of resources can be done using machine learning and the other important task like preprocessing and post-processing can be done using hand crafted rules.

3 Machine Learning for NE Recognition

3.1 Using Hidden Markov Models

Hidden markov models relies on three parameters that are a matrix A of tag transition probabilities, a matrix B of emission or observation probabilities and a matrix π in which probability of the tag to occur in the initial state are recorded. Trigram HMM is defined as (K, V, λ), where $K = \{s_1, s_2, ..., s_n\}$ is a finite set of possible states, $V = \{x_1, x_2, ..., x_n\}$ is a finite set of possible observations and $\lambda = (\pi, A, B)$, where, $\pi = \{\pi_i\}$: Set of initial state probabilities and π_i : Initial probability that system starts at state i, $A = \{a_{ij}\}$: Set of state transition probabilities and a_{ij}: Probability of going to state j from state i, $B = \{b_i\{x_k\}\}$: Set of emission probabilities and $b_i\{x_k\}$: Probability of generating symbol x_k at state i. Maximum likelihood estimates are used to estimate parameters of λ model as, $a_{ijk} = \frac{C(i,j,k)}{C(i,j)}$ and $b_i\{x_k\} = \frac{C(i \rightsquigarrow x_k)}{C(i)}$. The start of the sentence is marked by $**$ and end of the sentence is marked by $STOP$ tag. The probability of state sequence $s_1, s_2.....s_{n+1}$ for given $x_1, x_2.....x_n$ observation sequence for NE tagging can be computed as,

$$P(x_1x_2.....x_n|s_1s_2.....s_{n+1}) \cong \prod_{i=1}^{n+1} q(s_i|s_{i-2}, s_{i-1}) \times \prod_{i=1}^{n} e(x_i|s_i)$$

Where q and e are parameters for maximum likelihood estimates. If we have $n = 6, x_1, x_2, ..., x_6$ equal to the sentence टप्प्यात १० हजार रुपये अनुदान., and $s_1, s_2, ..., s_7$ equal to the tag sequence O B-AMT I-AMT E-AMT O O STOP, then Bigram counts (Mat_{BC}) for probable tag sequence O B-AMT I-AMT E-AMT O O STOP for the sentence टप्प्यात १० हजार रुपये अनुदान. is,

$\text{Mat}_{\text{BC}} =$

	*	B-AMT	I-AMT	E-AMT	O	STOP
*	26462	44	0	0	19544	0
B-AMT	0	0	937	491	0	0
I-AMT	0	0	768	936	0	0
E-AMT	0	24	0	0	1335	1
O	0	1227	0	0	265391	26305
STOP	0	0	0	0	0	0

Unigram counts (Mat_{UC}) for probable tag sequence O B-AMT I-AMT E-AMT O O STOP is

$\text{Mat}_{\text{UC}} =$

*	B-AMT	I-AMT	E-AMT	O	STOP
0	1428	1705	1427	323621	0

Bigram probabilities (Mat_{BP}) for probable tag sequence O B-AMT I-AMT E-AMT O O STOP is,

$$\text{Mat}_{\text{BP}} = \begin{array}{c} \\ * \\ \text{B-AMT} \\ \text{I-AMT} \\ \text{E-AMT} \\ \text{O} \\ \text{STOP} \end{array} \begin{array}{c} \begin{array}{cccccc} * & \text{B-AMT} & \text{I-AMT} & \text{E-AMT} & \text{O} & \text{STOP} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.656 & 0.344 & 0 & 0 \\ 0 & 0 & 0.450 & 0.549 & 0 & 0 \\ 0 & 0.017 & 0 & 0 & 0.936 & 0.001 \\ 0 & 0.004 & 0 & 0 & 0.820 & 0.081 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

The q and e parameter estimations for above sentence are

$$\begin{bmatrix} q(\text{O}|*) & = \frac{\text{C(*,O)}}{\text{C(*)}} & = 0.73857 \\ q(\text{B-AMT}|\text{O}) & = \frac{\text{C(O,B-AMT)}}{\text{C(O)}} & = 0.00379 \\ q(\text{I-AMT}|\text{B-AMT}) & = \frac{\text{C(B-AMT,I-AMT)}}{\text{C(B-AMT)}} & = 0.656162 \\ q(\text{E-AMT}|\text{I-AMT}) & = \frac{\text{C(I-AMT,E-AMT)}}{\text{C(I-AMT)}} & = 0.548974 \\ q(\text{O}|\text{E-AMT}) & = \frac{\text{C(E-AMT,O)}}{\text{C(E-AMT)}} & = 0.93553 \\ q(\text{O}|\text{O}) & = \frac{\text{C(O,O)}}{\text{C(O)}} & = 0.82007 \\ q(\text{O}|\text{STOP}) & = \frac{\text{C(O,STOP)}}{\text{C(O)}} & = 0.08128 \\ e(\text{टप्प्यात}|\text{O}) & = \frac{C(O \rightsquigarrow \text{टप्प्यात})}{\text{C(O)}} & = 0.000108 \\ e(\text{१०}|\text{B-AMT}) & = \frac{C(B-AMT \rightsquigarrow \text{१०})}{\text{C(B-AMT)}} & = 0.006303 \\ e(\text{हजार}|\text{I-AMT}) & = \frac{C(I-AMT \rightsquigarrow \text{हजार})}{\text{C(I-AMT)}} & = 0.226979 \\ e(\text{रुपये}|\text{E-AMT}) & = \frac{C(E-AMT \rightsquigarrow \text{रुपये})}{\text{C(E-AMT)}} & = 0.280308 \\ e(\text{अनुदान}|\text{O}) & = \frac{C(O \rightsquigarrow \text{अनुदान})}{\text{C(O)}} & = 0.000121 \\ e(.|\text{O}) & = \frac{C(O \rightsquigarrow .)}{\text{C(O)}} & = 0.079025 \end{bmatrix}$$

Bigram probability for an optimal tag sequence O B-AMT I-AMT E-AMT O O STOP for the sentence टप्प्यात १० हजार रुपये अनुदान. is,

$$\begin{aligned} P(x_1...x_6, s_1...s_7) = \\ & q(O|*) \\ & \times q(B-AMT|O) \\ & \times q(I-AMT|B-AMT) \\ & \times q(E-AMT|I-AMT) \\ & \times q(O|E-AMT) \\ & \times q(O|O) \\ & \times q(O|STOP) \\ & \times e(\text{टप्प्यात}|O) \\ & \times e(\text{१०}|B-AMT) \\ & \times e(\text{हजार}|I-AMT) \\ & \times e(\text{रुपये}|E-AMT) \\ & \times e(\text{अनुदान}|O) \\ & \times e(.|O) \\ & = 2.59683\times10^{-17} \end{aligned}$$

The probability of optimal tag sequence for a given word sequence is illustrated in above example. Similar probabilities are computed for all possible tag sequences for a given sentence using MLE estimation. Among all such possible tag sequences for a given sentence, the optimal path of tag sequence is to be selected. The tag sequence with highest probability is selected. This decoding is done by Viterbi algorithm(section 3.3). The trellis diagram for Viterbi decoding for a sample sentence टप्प्यात १० हजार रुपये अनुदान., is shown in figure 1.

3.1.1 Preprocessing Data

The lemmatization based technique (Patil (2017) is implemented in which inflected word forms are replaced by specialized tokens. Ontologies for number names in words, time, length, weight, electricity, temperature, area, volume and units of currency has been developed. The Marathi text is preprocessed using lemmatization based technique to deal with the inflections in named entities.

3.1.2 Minimizing Comparisons

Twelve different types of NEs using 40 tags need to be recognized by the NE recognizer. General trigram HMM assigns every tag to each word, computes bigram, trigram and unigram probabilities and assigns most probable

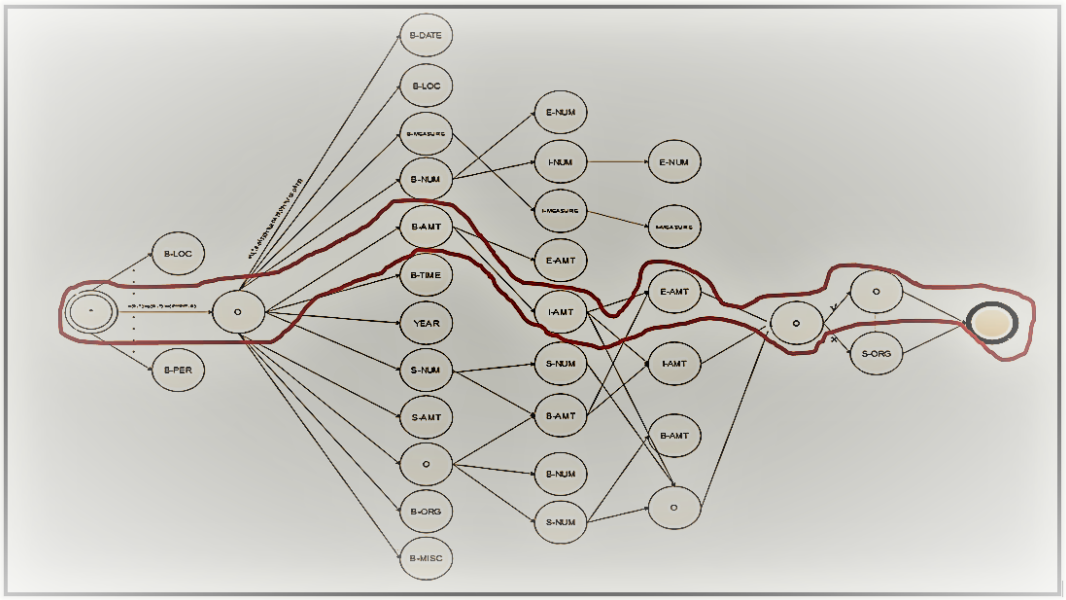


Figure 1: NE Tag Decoding

tag to the word based on maximum probability. We have taken two sentences to find number of trigram and emission probability computations. If 40 is the number of tags in tag set, then 40 tags are assigned to first word in sentence, 1600 bigram combinations are assigned to first and second word in sentence and 64,000 trigram combinations are assigned to first, second and third word. Thus, 64,000 trigram combinations are assigned to remaining words in sentence. Two * symbols are added to first word in sentence to make trigram. Trigram probability (T_P) is the ratio of trigram count (T_C) to bigram count (B_C) i.e. $T_P = T_C/B_C$. Combination B_C that is not seen in training becomes zero, zero value at denominator results in infinite trigram probability. The difficulty introduced because of unseen B_C is solved by using two solutions. First solution is return value 1 for unseen B_C which can temporary solve the problem. Second solution is find out all the bigram combinations which are never seen in training as well as not expected during testing. All such combinations are called invalid bigram combinations. There are approximately 1089 bigram tag combinations that are never seen in training and not expected in testing some of them are shown in table 1. For all combinations, which are invalid computation of T_P is skipped so that load of algorithm execution can be released to some extent as well wrong trigram state assignment to observations can be controlled. Comparison between first and second solution is shown in table 2.

CurTag	NextTag	CurTag	NextTag
B-LOC	B-TIME	B-LOC	I-DATE
B-LOC	B-AMT	B-LOC	I-MEAS
B-LOC	B-DATE	B-LOC	I-MISC
B-LOC	B-LOC	B-LOC	I-NUM
B-LOC	B-MEAS	B-LOC	I-ORG
B-LOC	B-MISC	B-LOC	I-PER
B-LOC	B-NUM	B-LOC	MONTH
B-LOC	B-ORG	B-LOC	O
B-LOC	B-PER	B-LOC	S-TIME
B-LOC	E-TIME	B-LOC	S-AMT
B-LOC	E-AMT	B-LOC	S-DATE
B-LOC	E-DATE	B-LOC	S-LOC
B-LOC	E-MEAS	B-LOC	S-MEAS
B-LOC	E-MISC	B-LOC	S-MISC
B-LOC	E-NUM	B-LOC	S-NUM
B-LOC	E-ORG	B-LOC	S-ORG
B-LOC	E-PER	B-LOC	S-PER
B-LOC	I-TIME	B-LOC	WEEKDAY
B-LOC	I-AMT	B-LOC	YEAR

Table 1: Part of Unseen Bigram Tag Combinations

T_P Computations	Solution 1	Solution 2
Trigram comparisons	142596	36581
Non zero T_Ps	3959	3959
Zero T_Ps	138637	32622

Table 2: Comparison between Tp Computations for Two Solutions

3.2 Viterbi Decoding

Viterbi algorithm is used to predict most likely tag sequence for an input sequence. The algorithm finds most probable state sequence $s_1, s_2, ..., s_n$ for a observation sentence $x_1, x_2, ..., x_n$. The problem of maximizing $P(s_1, s_2...s_n|x_1, x_2.....x_n)$ is addressed using $argmax_{s_1....s_n} P(s_1 s_2.....s_n|*, x_1 x_2.....x_n, STOP)$.

3.3 Handling Unseen words

Unseen words are absent in training, therefore their prediction probability becomes zero. If frequency of observation in test set is less than or equal to 5, then that observation is treated as rare word. Non frequent words in test set are replaced by $< RARE >$ token. Katz back-off smoothing is used to estimate the count of words that are never seen in training.

4 Linguistics for NE Recognition

Linguistic knowledge to recognize Marathi NEs is represented using indicator word lists, gazetteers, and grammar rules. This subsection provides brief information about the linguistic resources developed for detection of NEs from newspaper articles.

4.1 Indicator Word Lists

The indicators often surrounding the NEs can act as trigger words in identification of NEs in their context. Such words play significant role in designing heuristics to indicate NEs within the text. Certain words exist in text that are not indicators but are ambiguous NEs and must be treated separately. The word lists for indicators such as title person, awards, degree, person name suffixes, suffixes to person first name, suffixes to person last name, collision of proper and common noun, collision of proper, common noun and verbs, ambiguous last names, Marathi abbreviations, English in Devanagari abbreviations, location indicators, location suffixes etc. were developed to assists NE recognition by rule based NER algorithm.

4.2 Using Gazetteers

Gazetteer for first names, last names, organizations names, miscellaneous names, days of the week, month names (English and Marathi), single word location, organization and miscellaneous etc. were created. The word form(s) which is (are) untagged if found in some gazette(s), then the appropriate tag(s) is (are) assigned to the word form(s) based on the gazette(s) in which it found.

4.3 Using Grammatical Rules

The grammatical rules are a set of grammatical patterns designed to derive NEs based on lemmatization. Grammatical patterns were indicated using regular expressions. Several rules have been developed, which are used to extract person, location, amount, measure, date, time, and number entities.

5 Experimental Work

5.1 Dataset Preparation

FIRE-2010 corpus is used to develop NE annotated corpus by manually tagging 12 types of NEs. 27,177 sentences of Marathi text have been annotated using IOBES scheme. Training data developed for Marathi NER consists of 4,01,295 word forms that comprise of 12,303 person names, 7,440 organization, 10,015 location, 3,242 miscellaneous, 7,093 number, 1,500 amount, 2,967 measure, 1,549 date, 369 time, 197 month, 456 weekdays, and 395 year named entities. The rich morphology of the Marathi language allows adding suffixes and prefixes to a morpheme to add semantic to a word and to create meaningful context. It is observed during corpus annotation that almost all NE instances are present in inflected form. Although the dataset is large enough, frequency count of word is found to be lower since inflections result in same word appearing in different forms. This further results in poor probabilities and sparse data problem in MLE estimates. Lemmatization based preprocessing deals with such inflections and is used in the preprocessing of training and testing datasets.

5.1.1 Held Out Test Dataset Preparation

Two sets of training and testing datasets is created by dividing the NE annotated corpus preprocessed using lemmatization in 80:20 and 90:10 percent proportions. The actual number of sentences in the corpus are computed, 20% of the total sentences in the corpus were randomly selected and removed from the corpus. The set of randomly selected sentences

NE Class	NE Annotated Data	Training Dataset1	Held-out Dataset1	Training Dataset2	Held-out Dataset2
Person	12,303	11,998	0305	12,285	018
Organization	07,440	07,236	0204	07,421	019
Location	10,015	09,723	0292	09,983	032
Miscellaneous	03,242	03,170	0072	03,231	011
Number	07,093	06,893	0200	07,081	012
Amount	01,500	01,463	0037	01,494	006
Measure	02,967	02,887	0080	02,958	009
Date	01,549	01,515	0034	01,541	008
Time	00369	00360	0009	00363	006
Month	00197	00193	0004	00190	007
Weekday	00456	00441	0015	00455	001
Year	00395	00384	0011	00389	006
Total NEs	47,526	46,263	1,263	47,391	135
#Sentences	27,177	26,462	0715	27,127	050

Table 3: Held Out Training and Testing Dataset Details

is termed as Held-out dataset1. The remaining sentences (80%) in the corpus (training dataset1) were used to train the NER system. Similarly, 10% of the total sentences in the corpus were randomly selected, removed and stored in Held-out dataset2. The remaining sentences (90%) in the corpus (training dataset2) were used to train the NER system. The total number of NE instances found in the training dataset1, training dataset2, held-out dataset1 and held-out dataset2 are shown in table 3.

NE Class	Train1	Unseen1	Unseen2
Person	11,998	33	08
Organization	07,236	15	16
Location	09,723	17	22
Miscellaneous	03,170	16	02
Number	06,893	10	16
Amount	01,463	05	01
Measure	02,887	02	06
Date	01,515	03	03
Time	00,360	01	01
Month	00,193	02	01
Weekday	00,441	01	01
Year	00,384	04	04
Total NEs	46,263	109	81
Sentences	26,462	33	24

Table 4: Unseen Test Dataset Details

5.1.2 Unseen Test Dataset Preparation

Unseen dataset1 is a dataset composed of news items taken from online eSakal newspaper in October 2016. Unseen dataset2 is a dataset composed of news items taken from online eSakal newspaper in February 2017. Both the unseen datasets were tokenized and preprocessed using lemmatization. The total number of NE instances found in the unseen dataset1 and unseen dataset2 is shown in table 4. The NE annotated corpus pre-processed using lemmatization consisting of 27,177 sentences mentioned in the dataset preparation section is used to train the NER system.

5.2 NER System Architecture

The proposed NER system applies statistical algorithm i.e. trigram HMM using lemmatization algorithm to test data. This algorithm recognizes Marathi NEs satisfactorily. It also deals with unknown words and performs word disambiguation to some extent. There is possibility that some NEs might be untouched by the system. Therefore, rule and gazetteer based NER algorithm is cascaded to the NER system. The rule based algorithms do not modify the recognition carried by statistical algorithm, rather it tags only the untagged NEs in the test data. The NEs which are not contained in any gazetteer are termed as unseen NEs. The problem of unseen NEs is solved by statistical algorithm using pseudo word replacement. Therefore, continuous expansion of gazetteers is not required. Expected performance of the Marathi NE recognition is achieved using combining the statistical algorithm with the rule based algorithm. The architecture of NER system for the Marathi language that combines statistical named entity recognition, gazetteers and grammar rules is

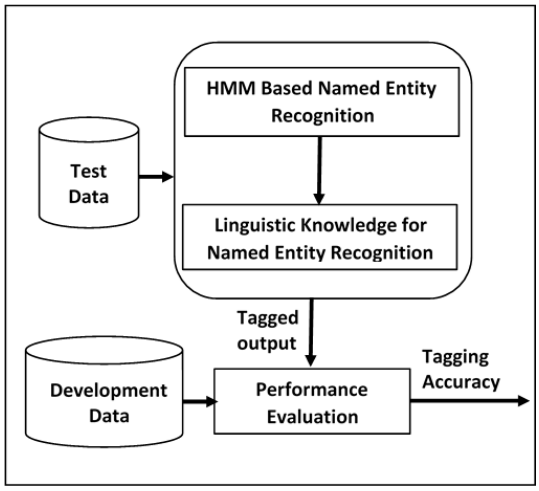


Figure 2: Marathi NER System

shown in figure 2.

5.3 Evaluation of Hybrid NER System

The performance of the Marathi NER based on hybrid approach is evaluated using four varying size datasets containing varying number of NEs. Out of them two datasets were held out and remaining two datasets were unknown datasets.
The system is trained on dataset(s) preprocessed using lemmatization. The performance of the system using held out datasets is shown in table 5 and 6. The overall NE identification accuracy reported by the system for held out dataset1 and 2 is 93.35% and 98.14% respectively. The average NE classification accuracy reported is 95.24% and 97.79% respectively.

The overall NE identification accuracy reported by the system for unseen dataset1 and 2 is 81.37% and 83.33% respectively which is relatively satisfactory. The average NE classification accuracy reported for unseen dataset1 and 2 is 83.09% and 84.23% respectively. The NE recognition accuracy for organization NE is relatively less result in unsatisfactory average NE classification accuracy for unseen dataset2. Numeric NEs in this dataset were accurately recognized than the enamex type of NEs by the system. The performance of the system using unseen datasets is shown in table 7 and 8 respectively. Overall NE identification accuracy and average NE classification accuracy is shown in graph 3 and 4 respectively.

NE Class	Precision	Recall	F1-Score
NEI	92.79	93.92	93.35
Person	84.05	86.35	85.18
Organization	95.02	98.96	96.95
Location	97.26	97.26	97.26
Miscellaneous	95.83	95.83	95.83
Number	96.43	90.43	93.33
Amount	80.00	100.0	88.89
Measure	100.0	100.0	100.0
Date	93.67	97.37	95.48
Time	81.82	100.0	90.00
Month	100.0	100.0	100.0
Weekday	100.0	100.0	100.0
Year	100.0	100.0	100.0
NEC	93.67	97.18	95.24

Table 5: NER System Performance on Held-out Dataset1

NE Class	Precision	Recall	F1-Score
NEI	98.51	97.78	98.14
Person	94.74	100.0	97.30
Organization	100.0	100.0	100.0
Location	100.0	96.88	98.41
Miscellaneous	100.0	100.0	100.0
Number	92.31	100.0	96.00
Amount	100.0	100.0	100.0
Measure	100.0	100.0	100.0
Date	100.0	100.0	100.0
Time	100.0	83.33	90.91
Month	100.0	100.0	100.0
Weekday	100.0	100.0	100.0
Year	100.0	83.33	90.91
NEC	98.92	96.96	97.79

Table 6: NER System Performance on Held-out Dataset2

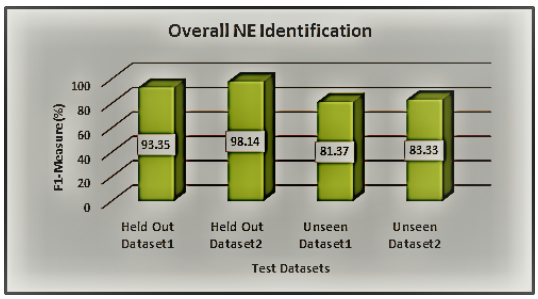


Figure 3: Overall NE Identification

NE Class	Precision	Recall	F1-Score
NEI	86.46	76.85	81.37
Person	78.57	66.67	72.13
Organization	90.91	66.67	76.93
Location	100.0	94.12	96.97
Miscellaneous	100.0	87.50	93.33
Number	69.23	90.00	78.26
Amount	66.67	80.00	72.73
Measure	100.0	50.00	66.67
Date	100.0	100.0	100.0
Time	100.0	100.0	100.0
Month	100.0	100.0	100.0
Weekday	100.0	100.0	100.0
Year	100.0	25.00	40.00
NEC	92.12	80.00	83.09

Table 7: NER System Performance on Unseen Dataset1

NE Class	Precision	Recall	F1-Score
NEI	92.31	75.95	83.33
Person	83.33	62.50	71.43
Organization	87.50	43.75	58.33
Location	85.00	77.27	80.95
Miscellaneous	100.0	0	0
Number	100.0	100.0	100.0
Amount	100.0	100.0	100.0
Measure	100.0	100.0	100.0
Date	100.0	100.0	100.0
Time	100.0	100.0	100.0
Month	100.0	100.0	100.0
Weekday	100.0	100.0	100.0
Year	100.0	100.0	100.0
NEC	96.32	81.96	84.23

Table 8: NER System Performance on Unseen Dataset2

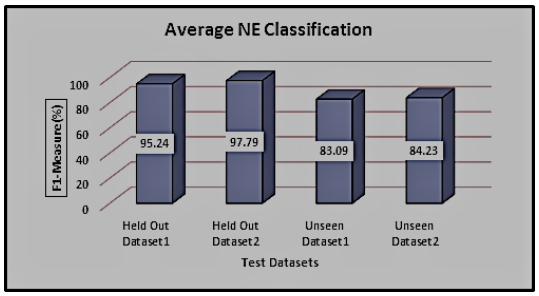


Figure 4: Overall NE Classification

The cumulative performance of Marathi NER system based on Hybrid approach for held out and unseen test datasets is shown in table 9. NE identification and classification reported by this system is 90% approximately, which is satisfactory for Marathi language.

Test Datasets	NEI	NEC
Held-Out Dataset1	93.35	95.24
Held-Out Dataset2	98.14	97.79
Unseen Dataset1	81.37	83.09
Unseen Dataset2	83.33	84.23
Average	89.05	90.09

Table 9: Average Performance of NER

6 Conclusion

A NER system for Marathi language is described that applies hidden markov model, language specific rules and gazetteers to the task of named entity recognition (NER) in Marathi language. Starting with named entity (NE) annotated corpora and lemmatization first a baseline NER system was implemented. Then some language specific rules are added to the system to recognize some specific NE classes. Also, some gazetteers and context patterns are added to the system to increase the performance. After preparing the one-level NER system, a set of rules are applied to identify the nested entities. The system can recognize 12 classes of NEs with 89.05% accuracy in average NE identification and 90.09% accuracy in average NE classification for held out and unseen test datasets in Marathi.

Acknowledgement

This research work is supported by grants under Rajiv Gandhi Science and Technology Commission, Govt. of Maharashtra, India.

References

Asif Ekbal and Sivaji Bandyopadhyay. 2007. *A Hidden Markov Model Based Named Entity Recognition System: Bengali and Hindi as Case Studies*. Springer International Conference on Pattern Recognition and Machine Intelligence (PReMI 2007) Heidelberg, LNCS, 4815:545–552.

Anup Patel, Ganesh Ramakrishnan and Pushpak Bhattacharya. 2009 *Incorporating Linguistic Expertise using ILP for Named Entity Recognition in Data Hungry Indian Languages.* In Proceedings of the 19th International Conference on Inductive Logic Programming (ILP'09), Leuven, Belgium,178–185.

Sudha Morwal, and Nusrat Jahan. 2013. *Named entity recognition using hidden markov model (hmm): An experimental result on Hindi, urdu and marathi languages.* International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE),3(4):671–675.

Praneeth Shishtla, Karthik Gali, Prasad Pingali, and Vasudeva Varma. 2008. *Experiments in Telugu NER: A Conditional Random Field Approach.* In Proceedings of the Workshop on NER for South and South East Asian languages (IJCNLP-08), Hyderabad, India,105–110.

P. Srikanth and Kavi Narayana Murthy. 2008. *Named Entity Recognition for Telugu.* In Proceedings of the Workshop on Named Entity Recognition for South and South East Asian Languages, Third International Joint Conference on Natural Langauge Processing (IJCNLP-08), Hyderabad, India, 41-50.

Krupka, G.R., and Hausman, K. 1998. *IsoQuest Inc: Description of the NetOwl Text Extraction System as used for MUC-7.* In Proceedings of Seventh Message Understanding Conference (MUC-7), Fairfax, Virgina.

William J Black, Fabio Rinaldi and David Mowatt. 1998. *Facile: Description Of The NE System Used For Muc-7.* In Proceedings of Seventh Message Understanding Conference (MUC-7), Fairfax, Virgina.

Awaghad Ashish Krishnarao, Himanshu Gahlot, Amit Srinet and D. S. Kushwaha. 2009. *A Comparative Study of Named Entity Recognition for Hindi using Sequential Learning Algorithms.* International Advance Computing Conference (IACC 2009), Patiala, India:1163-1168.

Kashif Riaz. 2010. *Rule-based Named Entity Recognition in Urdu.* In Proceedings of the 2010 Named Entities Workshop, ACL 2010, Uppsala, Sweden:126–135.

B. Sasidhar, P.M. Yohan, A. Vinaya Babu, A. Govardhan. 2011. *Named Entity Recognition in Telugu Language using Language Dependent Features and Rule based Approach.* International Journal of Computer Applications, 22(8):30-34.

Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. 1999. *An Algorithm that Learns What's in a Name.* Machine Learning, 34(1): 211-231.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. 2001. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data* In Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001):282-289.

GuoDong Zhou Jian Su. 2002 *Named Entity Recognition using an HMM-based Chunk Tagger.* In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, Pennsylvania:473-480.

Asif Ekbal and Sivaji Bandyopadhyay. 2008. *Bengali Named Entity Recognition Using Support Vector Machine.* In Proceedings of the Workshop on Named Entity Recognition for South and South East Asian Languages, Third International Joint Conference on Natural Langauge Processing (IJCNLP-08), Hyderabad, India: 51-58.

Raymond Chiong and Wang Wei. 2006. *Named Entity Recognition Using Hybrid Machine Learning Approach.* 5th IEEE International Conference Cognitive Informatics, (ICCI-2006), Volume 1:578-583.

Branimir T. Todorovic, Svetozar R. Rancic, Ivica M. Markovic, Edin H. Mulalic and Velimir M. Ilic. 2008. *Named Entity Recognition and Classification Using Context Hidden Markov Model.* 9th Symposium on Neural Network Applications in Electrical Engineering, NEUREL 2008, Belgrade, Serbia :43-46.

Alireza Mansouri, Lilly Suriani Affendey, and Ali Mamat. 2008. *Name Entity Recognition Approach.* International Journal of Computer Science and Network Security, 8(2):320-325.

Sitanath Biswas, S. Mohanty, S.P. Mishra. 2009. *A Hybrid Oriya Named Entity Recognition System: Integrating HMM with MaxEnt.* In Proceedings of 2nd International Conference Emerging Trends in Engineering and Technology (ICETET 2009), Nagpur:639-643.

Xueqing Zhang, Zhen Liu, Huizhong Qiu, Yan Fu. 2009. *A Hybrid Approach for Chinese Named Entity Recognition in Music Domain.* In Proceedings of Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing , Chengdu, China:677-681.

Nita Patil, Ajay Patil and B. V. Pawar. 2017. *HMM based Named Entity Recognition for inflectional language.* IEEE International Conference on Computer, Communications, and Electronics (COMPTELIX 2017):565-572.

Sentiment Analysis: An Empirical Comparative Study of Various Machine Learning Approaches

Swapnil Jain, Shrikant Malviya, Rohit Mishra, Uma Shanker Tiwary
Department of Information Technology
Indian Institute of Information Technology Allahabad
Allahabad-211012 (Uttar Pradesh)
{j.swapnil2050,shrikant.iet6153,rohit129iiita,ustiwary}@gmail.com

Abstract

The aim of this paper is to experiment with different machine learning approaches to predict/classify the sentiment on various available sentiment corpuses named as Subjectivity v1.0 corpus, IMDB movie review corpus, Rotten Tomatoes (RT) Movie Reviews corpus, Twitter sentiment dataset. Variants of Naive Bayes (NB) and Support Vector Machines (SVM) have been often used for text categorization as baseline. In this paper, we have tried to show that how embodying bigram and trigram features with Logistic Regression (LR), Mutinomial Naive Bayes (MNB) and Support Vector Machine (SVM) show significant improvent in the sentiment anlaysis. Another observation we obtained is that LR outperforms the MNB and SVM in both large as well as short (snippets) sentiment text when sentiment classes are limited to two/three. Furthermore, when the sentiment analysis task turn into a kind of multi-class classifiction instead of binary on large corpora, deep learning become dominant. We obtained testing accuracy of 96.6% and training accuracy of 98.8% on IMDB corpus by LR with unigram+bigram+trigram feature variant. Similarly, for Subjectivity v1.0 and twitter corpus, the same model returns better accuracy. But on the multi-class RT movie reviews corpus, Deep learning based proposed architecture-3 of type Extended-Convolution Neural Network (E-CNN) outperforms others.

1 Introduction

Recently, the field of Opinion Mining and Sentiment Analysis has enticed many researchers around the globe due to its capability of delivering valuable informative applications. People's opinion and reviews can play a crucial role in making decision's and choosing among multiple options when those choices are related on valuable resources for example expenditure of time and money to buy products as well as services. These information mostly sourced from *social web* through several forums, blogs and social networking websites. However, Due to its heterogeneous and unstructured nature, this information is not directly *machine processable*. Thus, it set the reason for the emergence of *Opinion Mining* (OM) and *Sentiment Analysis* (SA) as a prominent area of research. Both the keywords are commonly used interchangeably to denote the same meaning. However, some researchers believe, both aim to solve two slightly different problems. According to (Tsytsarau and Palpanas, 2012), OM determine whether a piece of text contains opinion or not, a problem that considered as *subjectivity analysis*. On the other hand, SA's task is to measure the polarity of text i.e. positive or negative.

Polarity classification is known to be very basic task of OM and SA. Polarity classification as the name signifies, classify a piece of text related to opinion on a particular issue into two sentimental opposite class. Moreover, it also helps in identifying pros and cons expressions of customer reviews which make the product evaluation and customer interest assessment more credible.

In the present scenario, sentiment analysis and opinion mining depends on the vector extraction of a piece of text in order to represent it's most salient and important features. These features representing a specific patterns-set help in determining the proper sentiment/opinion class. Term frequency, presence and tf-idf[1] are commonly used features.

[1]tf-idf, short for term frequency-inverse document frequency

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 112–121

In this research, we study the empirical effects related to several variants of LR, MNB, SVM on various available sentiment datasets. However, these approaches are already used enormously in text categorization, their performance varies due to inherent variability in features, datasets and model used. Through a set of experiments done on many datasets, we tried to show that the better selection of variants in many cases outperform the recent published state-of-the-art.

2 Related Work

Sentiment analysis field of research has been studied and employed widely since last two decades. SA systems have been implemented through different levels of analysis, such as word level e.g., (Qiu et al., 2009), the attribute level e.g., (Mei et al., 2007), the concept level e.g., (Cambria and Hussain, 2012), the sentence or clause level e.g., (Wilson et al., 2004) and finally the document level e.g. (Pang et al., 2002).

The Sentiment analysis is also understood as a task of determining the sentiment orientation of a given textual unit distinguished into two or more classes. Hence, the task of sentiment classification has also been implemented for different number of classes such as binary (e.g. positive/negative classification), ternary (e.g. positive/negative/neutral), n-ary (e.g. 1-5 star labelling) (Rui et al., 2013).

In general, the SA approaches can be classified into two main categories, the dictionary based approaches and other one is machine learning based approaches (Saad, 2014). Dictionary-based approaches are also known as lexical-based approaches that utilize a set of predefined set of sentiment dictionaries to identify the sentiments in a given text. At the starting, most of the work in the field of sentiment analysis was focused only on the dictionary-based approaches. On the other hand, machine learning approaches are become popular in recent years which work through constructing a classifier trained on manually annotated corpus to discriminate the sentiments of a given text.

Likewise, Decision Trees (DT), Naive Bayes (NB), Support Vector Machine (SVM), Neural Network (NN) and Maximum Entropy (ME) are the common set of supervised learning approaches, applied in sentiment classification (Medhat et al., 2014). Each type of approaches have its own pros and cons. For example, the dictionary-based approaches suffers from the limitation of highly domain-orientedness. Likewise, the machine learning approaches also require a significant human effort in order to annotate a substantial number of examples for training a classification model first.

OM and SA are in real, non-trivial and challenging problem, spanned over many areas and applications. However, a significant number of studies have been done in this field since past one decade, still much remained to be explored in order to build robust real-life applications. It has been observed that the problem of differentiating subjective with objective instances of sentiment is more difficult than the later polarity classification (Molina-González et al., 2013). Therefore, any improvement made on the field of subjective classification will put positive impact on sentiment classification. In the past, it has been done not only using machine learning (Wang et al., 2011; Pang and Lee, 2004) but lexicon-based approaches are also been adapted (Banea et al., 2014; Xuan et al., 2012). A glimpse of some subjective classification results obtained by the researchers in the past on Pang and Lee (Pang and Lee, 2004) corpus are shown in Table 1.

Sometimes, sentiment classification is related to identification of polarity of a piece of text whether it is showing positive, negative or neutral sentiment (Wilson et al., 2005; Turney, 2002). Therefore, sometimes sentiment classification is also called as polarity determination. Polarity determination has been tried on product reviews, blogs, micro blogs, news articles and forums. It's been observed that such texts are full of non-linguistic content e.g. abbreviations, noisy texts. Hence, it is required to use high level of preprocessing and more intelligent analytical techniques in order to extract most important discriminating patterns. These micro-blogs are proved to be more prominent and useful objects for many applications such as inferring opinion in social networks, twitter mood prediction, social advertising over micro-blogs and user-interest prediction in micro-blogging etc. (Maks and Vossen, 2012; Bollen et al., 2011; Bao et al., 2013; Li and Shiu, 2012).

3 Corpora Description

The goal of this paper is to deliver a comparative study of various machine learning approaches on different datasets. A number of relevant benchmark datasets are used and analysed with sev-

Authors	Data Split	Classifier Models	Cross Validation	Feature Selection	Baseline Accuracy (%)	Best Accuracy (%)
(Pang et al., 2002)	700 Positive 700 Negative	NB, ME, SVM	3-fold	unigrams presence	-	82.90
(Pang and Lee, 2004)	1000 Positive 1000 Negative	NB, SVM	10-fold	unigrams presence	87.15	87.20
(Mullen and Collier, 2004)	700 Positive 700 Negative	Hybrid SVM	10-fold	PMI,Turney, Osgood,Lemmas	83.50	87.00
(König and Brill, 2006)	1000 Positive 1000 Negative	Text Pattern + SVM, SVM	5-fold	unigrams bigrams	87.50	91.00
(Abbasi et al., 2008)	1000 Positive 1000 Negative	Genetic Algorithm Genetic Algorithm with SVM	10-fold	POS/Words n-grams Punctuation	87.95	91.70
(Prabowo and Thelwall, 2009)	1000 Positive 1000 Negative	Hybrid (Rule + Statistical and SVM)	5-fold	term frequency term presence	87.30	87.30

Table 1: Recently published Results in the literature on various versions of (Pang et al., 2002) movie review dataset.

Dataset	Type	No. of Textual Units	Positive	Somewhat Positive	Negative	Somewhat Negative	Neutral
Subjectivity v1.0 Corpus	Snippets of Movie Reviews	10662	5331	-	5331	-	-
IMDB Dataset	Movie Reviews	50k	25K	-	25K	-	-
Twitter Sentiment Dataset	Tweets on Flight Service	14640	2363	-	9178	-	3099
Rotten Tomatoes Dataset	Movie Reviews	156060	9291	32681	7565	27325	79198

Table 2: Statistics of the datasets used in this paper.

eral methods in order to find their individual characersitics towards the various approaches. We have considered four different corpora in order to perform the experiments: (1) Rotten Tomatoes Dataset (Kaggle-Competitions, 2017), (2) Subjectivity v1.0 Corpus (Pang and Lee, 2005), (3) IMDB Movie Review Dataset (Maas et al., 2011) and (4) Twitter Sentiment Dataset (Twitter-Crowdflower, 2017).

3.1 Rotten Tomatoes Dataset

This is one of the renowned corpus for statistical sentiment analysis on the collection of movie reviews prepared by Pang and Lee (Pang and Lee, 2004). The corpus[2] was prepared in order to classify movie reviews as positive or negative that are collected from the IMDB.com (Internet Movie DataBase). Initially, the corpus was consisted of 2000 full length reviews, 1000 each of positive as well as negative. Later, the dataset transformed to carry reviews of sentiments scaled in range [1-5]. Recently, a contest was hosted on (Kaggle-Competitions, 2017) with a huge corpus of movie reviews taken from rotten tomattoes on 5-star rating scale. We used this updated large corpora in this paper to see the the difference in results of various approaches. As the collected reviews are classified according to the rating system in terms of 5-star, mulit-class machine classification approaches are applied to develop a robust sentiment classification model.

3.2 Subjectivity v1.0 Corpus

A *sentence polarity dataset*[3] has been created by Pang & Lee, consists of 5331 of each positive as well as negative short movie reviews "snippets" (compulsorily one single long sentence) extracted from `www.rottentomatoes.com` (RT-s) (Pang and Lee, 2005). The aim of collecting this dataset is to understand the sentiment analysis paradigm on short subjective reviews and objective plot summaries instead of considereing the complete large reviews. Each snippet in the corpora is marked as "positive" if it is labelled "fresh" in `www.rottentomatoes.com` and the other snippets which are marked with "rotten" are considered to be negative reviews.

3.3 IMDB Review Dataset

Another movie review dataset has been collected by Andrew Maas at Stanford, sourced from IMDB (Maas et al., 2011). The dataset consists of 50,000 reviews in total, 25,000 of each positive as well as negative sentiments, conditioned on no more than 30 reviews from one movie. The reviews are distributed evenly into positive and negative classes so that the random selection will result in 50% accuracy. As movie reviews in IMDB are scored from 1 to 10 scale, the selected negative reviews are considered if its score is ≤ 4 out of 10 and for the positive reviews the threshold is set to ≥ 7

[2]The dataset is freely available at `www.cs.cornell.edu/people/pabo/movie-review-data/review_polarity.tar.gz`

[3]The dataset is freely available at `www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.tar.gz`

out of 10. Other reviews (neutral reviews) are not considered in this dataset.

3.4 Twitter Sentiment Dataset

This dataset originally came from crowd flowers library [4] (Twitter-Crowdflower, 2017). The dataset was generated through undertaking the sort of complaints received by each airline entirely by major U.S. air carrier customer service. The dataset includes tens of thousands of tweets as mentioned in the table 2, their respective carriers, the positive, negative, and neutral sentiment. This is a manually labelled corpus. In the process of corpus generation, users were asked to manually label the tweets as positive, negative or neutral with reasons of late flight, fast service etc.

4 Classification Models for Sentiment Analysis

4.1 Dataset Pre-processing & Feature Extraction

Data pre-processing is necessary task for sentiment analysis as it performs the process of cleaning and preparing text to be suitable as input to classification models (Haddi et al., 2013). Most of the sentiment dataset are made of the content extracted from websites e.g. Movie Reviews websites, product opinion websites, tweets from twitter etc. They all contain usually lots of noise and uninformative parts such as HTML tags, advertisements and scripts which needed to be removed before sending them for the classification. In order to prepare datasets for applying various machine learning approaches, we have designed a set rules for removal of the noise and uninformative parts i.e. HTML tags, rating indicators etc.

For all datasets, similar steps of pre-processing methods are undertaken. Following steps are followed for the same:

- Removing URL and getting data inside HTML Tag.
- Removing Repeating Characters, i.e. looove = love.
- Replacing emoticons with word happy and sad
  ':D' ':)' ':P' ';)' → happy
  ':(' ' ;(' ':—' → sad
- Replacing marks, ? → qmark , ! → exmark
- Removing stop words and replacing words like (don't → do not) or (thx/thnx → thanks) etc.

Feature Engineering is an important part of text analytics where features are extracted from text. First comes bag of words, a model where words are stored like the elements of a set with no word order or specific grammar known. Second is about use of different encodings. It states how the text could be represented in the form of vectors where the length of the vector is generally considered as the length of vocabulary i.e. the number of distinct words. First comes the very basic Count Encoding which is drawn from the frequency of a word, kept in the vector form. Similarly, the tf-idf encoding deals with constructing vectors of tf-idf weight of the words. Likewise, vector generation can also use ngrams and word-embeddings as features. Under ngram feature space, a single word is known as unigram, a sequence of two and three words are called bigrams and trigrams correspondingly.

Recently, word embeddings become top-notch in order to avail the use of dense or continuous vectors. Its main benefit arguably is that it does not require expensive annotation, instead it can be derived from large unannotated corpora that are readily available. Pre-trained embeddings can then be used in downstream tasks that use small amounts of labelled data. Various Transformations are there for use of word embeddings in a sentence i.e. mean transformation, image transformation. If each word in a sentence will have n embeddings, its mean transformation would be the mean of all the n embeddings. Thus, this will give rise to the feature vector of same length as the length of sentence. On the other hand, if we consider the length of embeddings n and feature vector length m, $[n * m]$ order can be considered as a gray scale image where every element represents pixel intensity and thus it can be feed into a convolution neural network or any other machine learning model as an image.

4.2 Support Vector Machines (SVM)

Support vector machines (SVM) has been applied in this work in order to classify the text units in a set of pre-defined sentiment classes. The algorithm got its name from the fact that it used to find those samples (support vectors) which find the widest frontier between the positive and negative samples in the feature space through demarcating

[4] The dataset is freely available at www.crowdflower.com/data-for-everyone/

those samples (support vectors). Due to its several advantages such as robustness in high dimensional space, versatility to any type of features, highly suitable for linear separable data and robust even when the data is sparsely distributed in the feature space, SVM become suitable to be applicable in many text categorization problems, motivated to be used in the SA. This has been proved by achieving good results on application of SVM in opinion mining and shown that it has overcome other machine learning techniques (OKeefe and Koprinska, 2009). A comparative study of several variants of SVM with other approaches are discussed briefly in the next section Experiments & Results.

4.3 Multinomial Naive Bayes (MNB)

Bayes Theorem based techniques that assumes independence among events/predictors are considered to be Naive Bayes approaches. In simple terms, one feature is not related to any other features, this is the general idea behind the working nature of Naive Bayes. Because of its less time complexity, this model is faster and can be easily used for large datasets. With the power of simplicity in hand, it is also known to outperform even highly complex classification models in many cases (Saad, 2014).

In the Multinomial variation of Naive Bayes, each textual data d is considered as a bag of tokens with each entry in it t_i representing the occurrence of a token or its tf-idf value or any other weight score (Wang and Manning, 2012). Therefore, d can be shown as a vector $\vec{x} =< x_1, x_2, ..., x_n >$, in which each x_i is bound to show the weight of t_i occured in d. Furthermore, each text unit d of a particular class c is considered to be the outcome of selecting individually $|d|$ tokens from F with replacement where each t_i has probability $p(t_i|c)$. Hence, $p(\vec{x}|c)$ is represented by following multinomial distribution:

$$p(\vec{x}|c) = p(|d|) \cdot |d|! \cdot \prod_{i=1}^{m} \frac{p(t_i|c)^{x_i}}{x_i!} \quad (1)$$

here, a common assumption is followed that $|d|$ does not depend on the class c. This is a method that has shown a significant improvement when combined with a combination of unigram, bigram and trigram.

4.4 Logistic Regression (LR)

We now look at the application of another algorithm for sentiment analysis named logistic regression (Wang and Manning, 2012). In terms of classifiers, logistic regression belongs to the exponential or log-linear classifiers family. Like other linear classifiers such as Naive Bayes, it also extracts a set of weighted features from the input, combining them linearly preceded by taking logs. In a more general way, logistic regression is represented by a classifier that classifies a data in two classes.

The most fundamental difference between Naive Bayes and Logistic Regression is that the Naive Bays is a generative classifier while the Logistic Regression is a discriminative classifier (Jurafsky and Martin, 2014). Naive Bays classifier is based on the concept that it probabilistically chooses which output label c is to be assigned to an input x through maximizing $p(c|x)$. It is perceived directly, Naive Bayes classifier used to estimate the best c indirectly on the basis provided likelihood $p(x|c)$ and prior class probability $p(c)$:

$$\hat{c} = \underset{c}{\operatorname{argmax}}\, p(c|x) = \underset{c}{\operatorname{argmax}}\, p(x|c)p(c) \quad (2)$$

Although LR differs in terms of estimating the probabilities, it is still similar to NB as being a linear classifier. LR estimates the term $p(c|x)$ through extracting a set of features from the provided input followed by fusing them linearly with weight vector (dot product) and then putting this combined value to a function. The beauty of exponential function for generating positive outcome, is used as being an applied function here. In general, the basic Logistic Regression formula for estimating the $p(c|x)$ is:

$$p(c|x) = \frac{1}{Z} exp\left(\sum_{i}^{N} w_i f_i(c, x)\right) \quad (3)$$

The denominator in the above equation Z is normalization factor which converts a exponent value to its probability. If vectors are represented by N values, the final equation of calculating the probability of x being of class c through LR:

$$p(c|x) = \frac{exp\left(\sum_{i=1}^{N} w_i f_i(c, x)\right)}{\sum_{c \in C} exp\left(w_i f_i(c', x)\right)} \quad (4)$$

A form of linear regression where the value which we want to predict i.e. c takes the discrete amount which further can be used as the label for a class. The cost function to estimate parameters for logistic regression for binary classification which we intend to minimize is given as follows:

$$J(f) = -\frac{1}{m}\Bigg[\sum_{i=1}^{m} c^{(i)} \log p(c^{(i)}|x^{(i)}) + (1-c^{(i)}) \log(1-p(c^{(i)}|x^{(i)}))\Bigg] \quad (5)$$

Where, m is the number of sample, x is the predictor. Since c here always belongs to either 0 or 1. The strategy used for multiclass classification we used is one versus all where only one class is considered while classification of the rest considered to be zero. Its been shown in the table later under Experiments & Results section that LR proved to be far better than MNB and SVM when it includes bigram and tigram based features.

4.5 Extended Convolution Neural Network (E-CNN)

In this section, we discuss a extended version of convolution neural network (E-CNN), a variant of the CNN architecture used by (Lan et al., 2016). We have deployed this multi-channel variant of CNN, E-CNN (Extended-CNN) in order to capture both semantic as well as sentiment information. In the E-CNN architecture, a sentence of length n with each contained word w_i represented by corresponding k-dimensional vector $w_i \in \mathbb{R}^k$ to the i-th word. Hence, a sentence of length n with added necessary padding if needed is supposed to be represented as

$$w_{1:n} = w_1 \oplus w_2 \oplus ... \oplus w_n, \quad (6)$$

here, $\oplus$ represents a binary operator of concatenating its two operands. Hence, the symbol $w_{1:n}$ refers to concatenated string of n vectors $w_1, w_2, ..., w_n$. Further, the convolution, a dot product operation, filters out a set of features and properties from the input through applying a $filter$ m $\in \mathbb{R}^{hk}$ window of size, say h words, where k is the dimension size of the word vector. In other words, the goal of convolution layer is to generate a feature map c ($c \in \mathbb{R}^{n-h+1}$) like $[c_1, c_2, ..., c_{n-h+1}]$ for input sentence s, where each term c_j is estimated through dot product of convolution filter m with h word vectors ending at word w_j (i.e., $w_{j-h+1:j}$):

$$c_j = f(m^T w_{j-h+1:j} + b) \quad (7)$$

where f is a non-linear activation function such as hyperbolic tangent (Tanh), rectified linear unit (ReLU) and $b \in \mathbb{R}$ is a biased term which allows the activation function to be shifted to left or right for successful learning. Likewise, all the filters convolutes individually to each possible window of the words in the sentence $w_{1:h}, w_{2:h+1}, ..., w_{n-h+1:n}$ in order to generate a $featuremap$

$$c = [c_1, c_2, ..., c_{n-h+1}] \quad (8)$$

Each filter tries to identify only one type of feature. Hence, in order to capture multiple features, CNN models generally employ multiple filters by varying the windows sizes or using the same filter with random initialization each time.

Further, in order to capture the necessary information from each feature map c, several $pooling$ methods have been presented such as averaged pooling (i.e., $\hat{c} = \frac{1}{h}\sum_{i=1}^{h} c_i$) or max-over-time pooling (i.e., $\hat{c} = max(c_i)$). We have used max-over-time pooling operation on the feature map in order to take the maximum value $\hat{c} = max(\mathbf{c})$ as a feature for the corresponding filter. Natural idea to use this pooling operation is to capture the most important feature, nothing but the highest value, for each feature map. The value obtained as features ($z = [\hat{c_1}, \hat{c_2}, ..., \hat{c_k}]$) after pooling are forwarded into a $softmax$ layer:

$$p(y = l|z; \theta) = \frac{e^{z^T.\theta_l}}{\sum_{k=1}^{K} e^{z^T.\theta_k}} \quad (9)$$

which estimates the probability distribution over predefined labels l. Nevertheless, in order to adjust the weights of layers, the parameters in CNN model (i.e. m, f, b, θ) are fine-tuned via back-propagation method.

An experiment is done with two *channels word vectors*, one's job is to capture unsupervised semantic information and another's task is to extract sentiment details from the input. The first, semantic channel is kept static throughout the training and second, sentiment channel is fine-tuned via back-propagation (Kim, 2014).

Model	Training Accuracy	Testing Accuracy
LR Unigram	92.5	91.4
LR Bigram	88.9	87.5
LR Unigram + Bigram	93.3	92.6
LR Unigram + Bigram + Trigram	**98.8**	**96.6**
NB Unigram	91.3	90.2
NB Bigram	92.3	90.3
NB Unigram + Bigram	93.2	90.2
NB Unigram + Bigram + Trigram	**94.4**	**93.6**
SVM + Unigram	85.3	83.4
SVM + Bigram	79.0	77.5
Mean Embeddings + SVM	85.2	84.5
Mean Embeddings + LR	84.2	83.0

Table 3: Accuracy chart of various approaches on IMDB corpus.

5 Experiments & Results

Support Vector Machines are used with different kernels for classification and also in Logistic Regression; we use regularization to penalize the weights to prevent over-fitting. For E-CNN different approaches are taken like changing the filter sizes, pooling layers are also used, changing the number of hidden layers etc. Among various architectures of convolution networks major ones which give promising results, are listed as follows:

- **Architecture-1**
  - Convolution Layer 1D Receptive Field 3x1, Feature Maps 100, Activation relu
  - Convolution Layer 1D Receptive Field 4x1, Feature Maps 100, Activation relu
  - Max Pooling Layer 1D Receptive Field 3x1, Activation relu
  - Fully Connected Layer Neurons 100, Activation relu
  - Fully Connected Layer Neurons 50, Activation sigmoid
  - Output Layer
- **Architecture-2**
  - Convolution Layer 1D Receptive Field 2x1, Feature Maps 150, Activation relu
  - Max Pooling Layer 1D Receptive Field 3x1, Activation relu
  - Convolution Layer 1D Receptive Field 3x1, Feature Maps 150, Activation relu
  - Max Pooling Layer 1D Receptive Field 3x1, Activation relu
  - Fully Connected Layer Neurons 200, Activation relu
  - Fully Connected Layer Neurons 100, Activation sigmoid
  - Fully Connected Layer Neurons 50, Activation sigmoid
  - Output Layer
- **Architecture-3**
  - Convolution Layer 2D Receptive Field 3x3, Feature Maps 100, Activation relu
  - Convolution Layer 2D Receptive Field 3x3, Feature Maps 150, Activation relu
  - Flatten Layer
  - Fully Connected Layer Neurons 100, Activation relu
  - Fully Connected Layer Neurons 100, Activation relu
  - Fully Connected Layer Neurons 64, Activation relu
  - Fully Connected Layer Neurons 10, Activation sigmoid
  - Output Layer
- **Architecture-4**
  - Convolution Layer 2D Receptive Field 5x5, Feature Maps 100, Activation relu
  - Max Pooling Layer 2D Filter Shape 2x2
  - Convolution Layer 2D Receptive Field 4x4, Feature Maps 150, Activation relu
  - Max Pooling 2D Filter Shape 2x2
  - Flatten Layer
  - Fully Connected Layer Neurons 100, Activation relu
  - Fully Connected Layer Neurons 64, Activation sigmoid
  - Output Layer

On the basis of various feature combinations, many possible variants of SVM, MNB and LR such as have been investigated, but only those are mentioned in the tables 3, 4, 5 and 6 which deliver good results. As per the experiments done on the IMDB movie review corpus, It is observed that combination of Unigram, Bigram and Trigram features provide more accurate classification results. Table 3 supports the observation. Both Classes positive/negative are well-classified, but the sentences which were misclassified are mostly related to sarcasm or confusing for human perception. For example -

Predicted Negative but Marked Positive → "You are a total idiot if u dont watch this movie. You are wasting your time on this planet." (Sarcasm)

Model	Training Accuracy	Testing Accuracy
Logistic Regression Unigram	91.9	90.7
Logistic Regression Bigram	82.6	80.9
Logistic Regression Unigram + Bigram + Trigram	**99.7**	**96.4**
Naive Bayes Unigram	94.2	92.9
Naive Bayes Bigram	86.7	84.9
Naive Bayes Unigram + Bigram + Trigram	**97.1**	**95.2**
SVM + Unigram	82.4	82.1
SVM + Bigram	56.2	55.8

Table 4: Accuracy chart of various approaches on Subjectivity v1.0 corpus.

Model	Training Accuracy	Testing Accuracy
Naive Bayes + Count Encoding + unigrams + bigrams	80.3	5-fold
Naive Bayes + Tf-Idf Encoding + unigrams + bigrams	75.3	7-fold
Logistic Regression + Count Encoding + unigrams + bigrams	**82.1**	**5-fold**
Logistic Regression + Tf-Idf Encoding + unigrams + bigrams	81.5	7-fold

Table 5: Avg. Cross Validation Accuracy chart of various approaches on Twitter dataset.

Positive but Marked Negative → "This movie makes me wonder what I am doing on earth wasting time, doing nothing, Ohh Man, What the hell." (Confusing even for human perception)

With reference to published results on subjective v1.0, a sentiment corpus consists of snippets, short reviews, the results presented in this paper is more accurate. Moreover, it also shows, the capability of SVM is better in classification of long reviews. But for the short reviews or snippets as subjective corpus, Logistic regression and Naive Bayes are more accurate and robust. Addition of bi-grams improves the performance significantly as shown in Table 4. After the inclusion of trigram again improve the performance a bit more. Both LR and MNB with unigram, bigram and trigram features provides 96.4% and 95.2% accuracy respectively as shown in Table 4.

For the sentiment analysis experiment on twitter corpus, a number of encoding considered to draw feature set in order to apply some supervised learning methods. Here also, feature vectors are constructed out of various possible combination of unigrams, bigrams with individual count encoding and tf-idf encoding. Out of all combination, only those are shown here which draw significantly better result. The accuracy measurement is done on the set environment of 5-fold and 7-fold avg. cross-validation. The overall average accuracy is obtained 82.1% as shown in the Table 5 through logistic regression in combination with count encoding and unigram+bigram.

The Rotten Tomato Dataset is a very large movie review corpus composed of 156,060 sentences rated under 5-star rating scheme in Negative, Somewhat Negative, Neutral, Somewhat Positive, Positive categories. We divided the overall corpus into a ratio of 7:2:1 for training, test and cross-validation set. The problem of sentiment analysis now turned from binary classification to multi-category classification which make it difficult for the above implemented models to be incorporated here. This is the reason, the accuracy of some linear approaches such as SVM and LR start declining. Therefore, deep learning is undertaken to see the difference. Four different architectures are devised empirically which show better accuracy compare to SVM and LR as shown in Table 6.

It is clear from the above discussion that logistic regression works better on the datasets like imdb, subjectivity v1.0 and twitter where the sentiment classes are limited to two/three and the corpus is build of short statements/reviews. But for the large datasets like Rotten Tomato corpus which consists of millions of texts divided into many sentiment classes, a better model is required robust enough to capture and support entire feature set necessary

Model	Training Accuracy	Testing Accuracy
Mean Embeddings + SVM	52.4	52.0
Count Encoding + LR	61.4	60.5
TfIdf Encoding + LR	63.3	62.5
Architecture-1	58.3	56.2
Architecture-2	59.1	57.5
Architecture-3	**68.4**	**66.7**
Architecture-4	65.1	63.4

Table 6: Accuracy chart of various approaces on Rotten Tomatoes Dataset.

for the classification. For the twitter dataset, only unigrams and bigrams with count encoding give the better results.

As it can be seen, mean transformation of embeddings does not play major role in sentiment analysis whereas image transformation of embeddings achieve the best result among all other classifiers. It is not worth denying that mean transformation is not that good for representation of embeddings as feature vector. Many other transformations for embeddings are there like median, mode, tf-idf but still the combination of convolution neural network with image transformation of embeddings beats them all. So embeddings are quite useful if used wisely.

6 Conclusion

In this paper, we performed a set of experiments to capture the residing variation in various sentiment datasets such as short or long texts and binary vs multi-class classification variations. For this, we analyzed various renowned models for classification and also various architectures for Convolutional Neural Network on all possible datasets ranging from short reviews/snippets to long documents. For each type, a list of best performing models are shown. We observe that for short texts and/or binary classification LR models beat all other models with certain features. In contrast, for long texts like the rotten tomatoes datasets, logistic regression is shown to give accuracy of 62.47% but in order to achieve better accuracy we included the use of word embeddings as an image and feed it into the convolution neural network (proposed architecture-3) where we achieve greater accuracy of 66.70%. Furthermore, for multi-class dataset like rotten tomatoes dataset, based on the analysis of confusion matrix, a better feature set selection and corresponding model enhancement related problems can be considered for future work.

References

Ahmed Abbasi, Hsinchun Chen, and Arab Salem. 2008. Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. *ACM Transactions on Information Systems (TOIS)* 26(3):12.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2014. Sense-level subjectivity in a multilingual setting. *Computer Speech & Language* 28(1):7–19.

Hongyun Bao, Qiudan Li, Stephen Shaoyi Liao, Shuangyong Song, and Heng Gao. 2013. A new temporal and social pmf-based method to predict users' interests in micro-blogging. *Decision Support Systems* 55(3):698–709.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science* 2(1):1–8.

Erik Cambria and Amir Hussain. 2012. *Sentic computing: Techniques, tools, and applications*, volume 2. Springer Science & Business Media.

Emma Haddi, Xiaohui Liu, and Yong Shi. 2013. The role of text pre-processing in sentiment analysis. *Procedia Computer Science* 17:26–32.

Dan Jurafsky and James H Martin. 2014. *Speech and language processing*, volume 3. Pearson London.

Kaggle-Competitions. 2017. Kaggle: Sentiment analysis on movie reviews. https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data/. [Online; accessed 22-March-2017].

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Arnd Christian König and Eric Brill. 2006. Reducing the human overhead in text categorization. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 598–603.

Man Lan, Zhihua Zhang, Yue Lu, and Ju Wu. 2016. Three convolutional neural network-based models for learning sentiment word vectors towards sentiment analysis. In *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, pages 3172–3179.

Yung-Ming Li and Ya-Lin Shiu. 2012. A diffusion mechanism for social advertising over microblogs. *Decision Support Systems* 54(1):9–22.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages

142–150. http://www.aclweb.org/anthology/P11-1015.

Isa Maks and Piek Vossen. 2012. A lexicon model for deep sentiment analysis and opinion mining applications. *Decision Support Systems* 53(4):680–688.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal* 5(4):1093–1113.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*. ACM, pages 171–180.

M Dolores Molina-González, Eugenio Martínez-Cámara, María-Teresa Martín-Valdivia, and José M Perea-Ortega. 2013. Semantic orientation for polarity classification in spanish reviews. *Expert Systems with Applications* 40(18):7250–7257.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *EMNLP*. volume 4, pages 412–418.

Tim OKeefe and Irena Koprinska. 2009. Feature selection and weighting methods in sentiment analysis. In *Proceedings of the 14th Australasian document computing symposium, Sydney*. Citeseer, pages 67–74.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 271.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 115–124.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, pages 79–86.

Rudy Prabowo and Mike Thelwall. 2009. Sentiment analysis: A combined approach. *Journal of Informetrics* 3(2):143–157.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2009. Expanding domain sentiment lexicon through double propagation. In *IJCAI*. volume 9, pages 1199–1204.

Huaxia Rui, Yizao Liu, and Andrew Whinston. 2013. Whose and what chatter matters? the effect of tweets on movie sales. *Decision Support Systems* 55(4):863–870.

Farag Saad. 2014. Baseline evaluation: an empirical study of the performance of machine learning algorithms in short snippet sentiment analysis. In *Proceedings of the 14th International Conference on Knowledge Technologies and Data-driven Business*. ACM, page 6.

Mikalai Tsytsarau and Themis Palpanas. 2012. Survey on mining subjective data on the web. *Data Mining and Knowledge Discovery* 24(3):478–514.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 417–424.

Twitter-Crowdflower. 2017. Crowdflower: Sentiment analysis on twitter dataset. `https://www.crowdflower.com/data-for-everyone/`. [Online; accessed 22-March-2017].

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, pages 90–94.

Suge Wang, Deyu Li, Xiaolei Song, Yingjie Wei, and Hongxia Li. 2011. A feature selection method based on improved fishers discriminant ratio for text sentiment classification. *Expert Systems with Applications* 38(7):8696–8702.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*. Association for Computational Linguistics, pages 347–354.

Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. 2004. Just how mad are you? finding strong and weak opinion clauses. In *aaai*. volume 4, pages 761–769.

Huong Nguyen Thi Xuan, Anh Cuong Le, and Le Minh Nguyen. 2012. Linguistic features for subjectivity classification. In *Asian Language Processing (IALP), 2012 International Conference on*. IEEE, pages 17–20.

Handling Multi-Sentence Queries in a Domain Independent Dialogue System

[1]**Prathyusha Jwalapuram** [2]**Radhika Mamidi**
Language Technology Research Center,
Kohli Center on Intelligent Systems,
International Institute of Information Technology,
Hyderabad, India
[1]prathyusha.jwalapuram@research.iiit.ac.in
[2]radhika.mamidi@iiit.ac.in

Abstract

This paper discusses the handling of multi-sentence queries in a mixed-initiative dialogue system based on a hierarchically structured knowledge base, in a way that is domain independent. The system is rule-based and uses dependency relations and part-of-speech tags obtained from the Stanford Parser coupled with the hierarchical structure of the knowledge base to identify the user's goal. The system was tested for its accuracy over answering questions, and also subjective testing was done to evaluate the dialogue flow; primarily over the books domain. We show examples of the system developed over the domains of books, movies and restaurants to demonstrate the domain independence.

1 Introduction

Most dialogue systems focus on processing the user's input and classifying the dialogue in terms of the amount of information it presents and the possible paths the dialogue could take. The idea is to predict a possible goal of the user in order to be able to ask relevant questions if needed in order to fill information gaps, and provide more relevant replies. Using a hierarchically structured knowledge base for a dialogue system helps achieve this. They help us limit the possible paths of the dialogue, and can help us identify irrelevant inputs or topic changes.

Few dialogue systems attempt to process multi-sentence inputs (multiple utterances in a single user turn) that collectively behave as a query. Users of a dialogue system may break up their queries into multiple sentences, and also provide additional information and qualify their initial statements. In such cases it might not suffice to simply find all the relevant keywords; it might be important to understand the relationships between them as well.

Aided with dependency relations, part-of-speech tags (provided by the Stanford parser) and the inherent semantics of some words such as 'and' or 'but', we attempt to link the keywords in complex sentences and multiple sentences to get a clear picture of exactly what the user is looking for. Significant information can be gained by looking at function words and their dependents. We also attempt to look at expressions of negation and negative words indicating the exclusion of certain objects as required by the user.

The system is domain-independent, and we present an implementation over a simple, handcrafted knowledge base of books, movies, and restaurants, that is essentially structured into a useful hierarchy. We evaluate the system based on whether the replies are relevant or not on multi-sentence queries that were collected through a survey (objective evaluation) and qualitatively through participant interaction and rating (subjective evaluation).

The paper is organised as follows. Section 2 describes related work, section 3 describes the strucure of the knowledge base, section 4 describes the dialogue manager, section 5 shows some examples of dialogue demonstrating domain independence, section 7 is about the evaluation and error analysis and section 8 explores possible future work and the appendix at the end has more examples of dialogue with multiple-sentence inputs.

2 Related Work

The advantages of using an ontological knowledge base for a dialogue system were put forth in Milward and Beveridge (2003) which defined an ontology simply as a network of concepts and

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 122–130

instances related to each other through semantic links. They introduce a mixed-initiative dialogue system that uses part-whole and is-a relationships to drive clarification questions and determine the sequence of the dialogue. Since we seek to further simplify the structure into a more generic relationship set (rather than is-a and part-whole relationships), we refer to our knowledge base as hierarchically structured.

It has been argued that the separation of the dialogue management from the domain knowledge management helps reduce the complexity of the systems and enhance further extensions (Flycht-Eriksson, 2004). Flycht-Eriksson (2004) uses is-a and part-of relations to resolve issues of under and over specification. We use a similar approach to make our implementation domain independent; swapping the knowledge base with another in the same format would produce a working dialogue system for the new knowledge base, even of a different domain.

Dzikovska et al. (2003) describe a system that maintains two ontologies, a domain independent ontology for a parser linked to the lexicon to capture the aspects of dialogue interaction that are common across domains and a domain-specific ontology for knowledge representation, and they integrate the linguistic and domain knowledge by defining a set of mappings between the two.

Division of the task and the dialogue is also explored in Bohus and Rudnicky (2003). A dialogue engine generates domain independent conversational behaviors and the dialogue task specification is handled separately. RavenClaw uses hierarchical task decomposition, which has a tree-like structure. The dialogue task specification is domain specific.

Lee et al. (2009) propose an example-based dialogue modeling that is applicable over different domains, but requires dialogue corpora for each domain; they also do not consider multi-sentence utterances in the course of a dialogue.

Mazuel and Sabouret (2006) propose a generic command interpreter for natural languages that uses an ontology to clarify semantic concepts, coded in a specific language. They use a tokenizer, tagger, lemmatizer and a chunker, but discount the need for a grammar based syntactic parser; they remove stop words and treat the sentences as a bag of words, and use the ontology for concept matching and semantic analysis; they do not take into account the semantic information that is provided by function words such as 'in', 'and' and so on. They assume that the users' commands are unlikely to be complete sentences, and therefore do not consider multiple sentences at all.

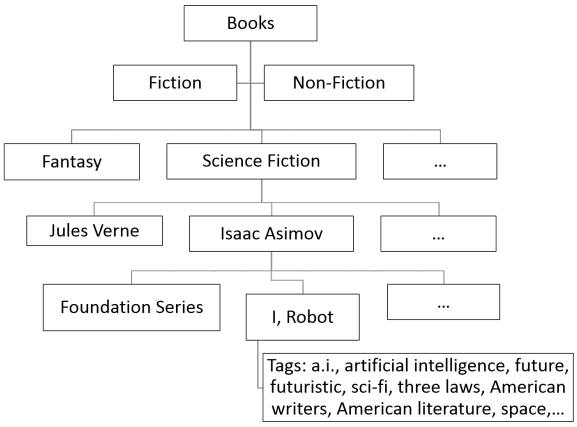


Figure 1: Hierarchical Knowledge Base Structure

Bickmore et al. (2011) describe an ontology based dialogue system that simulates a health counselor, using an RDF-based ontology described in OWL. They maintain a representation of a plan tree, recording the recipes that are actively being used and their goal-subgoal task decomposition relationships. Their dialogue fragments are modeled through a task representation language, and a dialogue planner enacts these fragments. They introduce the notion of adjacency pairs which are logically related, consisting, for example, of an utterance and its response.

Bharati et al. (1995) proposed a Computational Paninian Grammar framework for interpreting natural language queries, for example by creating verb frames of a list of domain-specific verbs in order to identify relationships between the keywords. We try to make it a domain independent implementation by using dependency relations with less significance given to the verbs, allowing us to focus on the relationships between the prepositions, adjectives, nouns/noun phrases, etc.

3 Hierarchically Structured Knowledge Base

The knowledge base is structured in a hierarchical manner for ease of representation, and also to facilitate dialogue flow. For example, in the *books* domain, the books maybe in a hierarchy from *genre* to *author* to *title* and so on. This helps

form a possible path that a dialogue may take; for example the user may specify a *genre* they like, and the system may suggest relevant *authors*; the user may then pick an author and the system suggests *books* by the author, and so on.

Another reason for maintaining a structured knowledge base rather than a standard ontology is the flexibility in defining relationships. In our knowledge base, the entities are related by a *'is-x-of'* relationships; that is, *is-author-of*, *is-genre-of*, etc. This could allow us to structure knowledge bases in domains which may not fit into the traditional ontology structure. We can also define synonyms for the relationships for making search easier *(author, writer)*.

Tag Information

Although typical data about books consists of information like author, year of publishing, genre, etc., in order to answer queries, the system must also be to consider what the books are about. This is problematic since we usually do not have information about the content of the books; the titles of the books are not always informative, and a large set of questions would have to go unanswered.

To resolve this issue and expand the scope of queries being answered, tags provided for books by the general public on popular book sites were collected and added to the knowledge base (or alternatively, high-frequency content words from book reviews can also be scraped). Tags are commonly used in searches for a similar purpose, allowing you to look for a book based on themes, characters, and other classifications which are not necessarily genres or part of a typical knowledge base.

Consider a query like *books with magic and animals in them*. The *Harry Potter* series, for example, would be a good fit for such a query; but nothing in any of the titles of the series would directly imply this. However, some of the tags for the series include *dark magic, witches, wizards, young adult, British fiction, beasts, dragons*, etc, which would allow us to infer that both *magic* and *animals* play a part in these books (synonyms must be considered).

Similarly, *women writers, female writer* are part of the tags for both the *Harry Potter* series and the *Hunger Games* series; this helps us return results to queries like *Do you have books written by female authors?*. All additional information, like prizes won by the books, the time period it is set

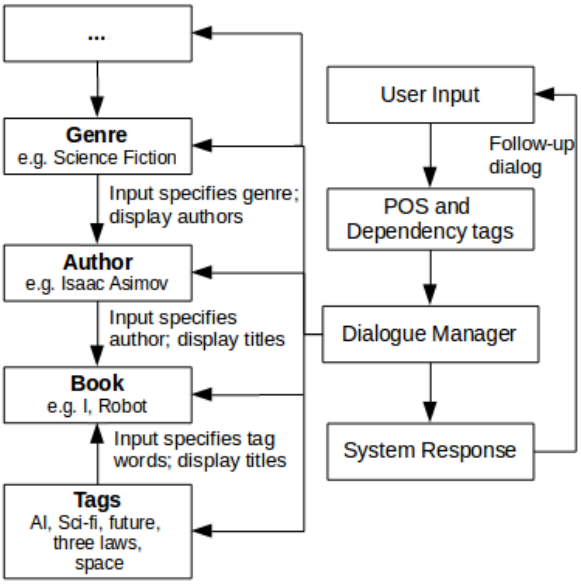


Figure 2: Simplified System Architecture

in (for example, *detective fiction set in the Victorian times = Sherlock Holmes*), etc. is available to us through these tags, enabling us to provide results to a wide variety of requirements; including spelling variations (*theatre, theater*).

For the movie domain, similar *'tags'* are available in terms of *'plot points'*, or high-frequency content words can be scraped from movie reviews on popular movie review sites. For the restaurant domain, we use the restaurant menu item descriptions (which are mainly descriptions of ingredients and the like).

4 Dialogue Manager

Our emphasis was to try and identify the objective of a user's query, that is, the data the user is looking for and the constraints pertaining to this data. We use the Stanford POS Tagger and the Stanford Dependency Parser for this purpose while assuming that the queries are free of errors and are fairly grammatically sound. We don't attempt to resolve abbreviations and ambiguities. The system is able to extract information from ungrammatical queries in certain cases where the Stanford Parser is able to generate a fairly accurate dependency parse.

We consider the cases where the user directly specifies what they are looking for, or makes oblique remarks that are intended to help reach the goal. These statements can be simple, complex or span multiple sentences.

An online survey was conducted to obtain questions for the development and testing. A total of 57

participants submitted 118 questions. All domain specific assumptions and rules are based on a development set of 68 questions; the other 50 were separated for testing.

4.1 Motivation behind using Dependency Relations

The advantage of using dependency relations is that they are syntacto-semantic relations, so the same question formulated in different ways can lead to the similar dependency relations (such as a statement in active/passive voice); this allows us to easily group similar user queries without having to anticipate all the possibilities.

Dependency relations also give us an idea of the relationship between the words and therefore the information the user is looking for and the constraints. This information cannot be obtained from simple syntactic parsing, such as Phrase Structure Trees, as they are highly dependent on the word order.

4.2 Processing the parser output

Keywords and Negation

An example of an input by the user could be *I like Oscar Wilde*. Here, the parser should tag *like* as the root, *I* as the nsubj (subject) and *Oscar Wilde* as the dobj (direct object). We see that the object here is the keyword we are looking for. Using such information, keywords in the utterance can be detected (Jwalapuram and Mamidi, 2017).

Similarly, we also consider cases where the user expresses a negative sentiment, such as *I don't like Oscar Wilde*. Here the negation applies to the *root*, and *Wilde* is the object of the root, so we transfer the negation from the root to the keyword; we remember to eliminate *Oscar Wilde* from any results we provide to the user. We also recognize lexicalized negatives, such as *hate, dislike, awful, stupid* etc. through a simple dictionary list of negative emotions. By maintaining a list of rejected candidates, we prune the results tailored to the user's requirements.

Prepositions, Modifiers and Conjunctions

In order to maintain domain independence, we cannot attribute specific interpretations to function words, which are often overloaded. We use prepositions as a clue merely signalling relationships between two keywords (nouns or verb and noun). This relationship does not need to be identified; we use the knowledge base to detect this. For example, if the user says *"I want books by Dan Brown"*, we identify that *"books"* and *"Dan Brown"* are the keywords, and are related. A reference to a topic in *'What books on psychology are available?'* is identified in a similar way, as related keywords *"books"* and *"psychology"*. Useful modifications through adjectives include cases like *I want funny books* or *Do you have scary books?*.

And and *or* are treated somewhat similarly. If a user says *I like magic and animals*, if there are no books which are about both *magic* and *animals*, we would still want to return some results, which are about magic or animals. This would be equivalent to the user saying *'I like magic or animals'*; in this case, all books which have references to either *magic* or *animals* are part of the results (see also Table 11 in Example Dialogs). For *but*, it is a clear case of contrast; any keyword obtained from a *but* clause are included in the reject list. Consider *I want books about vampires but not Twilight* or *All books by Bernard Shaw but Pygmalion*.

Relative Clauses Consider *I would prefer books which have some mystery in them*. Here the user is looking for *books*, which is the argument of the main verb *prefer* and the user specifies that the books need to *have* (secondary verb) *mystery* (object of secondary verb). Note that the prepositional argument here is *them*; since it is a demonstrative pronoun, it can be discarded as irrelevant. Using dependency relations, it is possible to extract the information that our keywords *books* and *mystery* are therefore related (see Table 8 in Example Dialogues).

In case of *but*, the same assumption of contrast applies; we simply identify the relevant keyword as a reject qualification and eliminate it from the results (*I would like books which are about traveling but not those which are focused on Europe*).

4.3 Multiple Sentences

We have a similar assumption here, that the user is likely to state their primary requirement first, followed by further qualifications. Since we add any keywords which have been associated with negative words to a reject list, and simply add any other keywords to a list of possibilities, the explicit order does not really matter. However, it is assumed that the set of sentences which occur together are related; which means that any qualifications that are made are applied collectively.

Consider *I like books with humor. I don't like*

Me: I like science fiction. Maybe Asimov.
System: I found these under Science Fiction: *H. G. Wells* *Jules Verne* *Isaac Asimov* *George Orwell* *Aldous Huxley* I found these under Isaac Asimov: *I, Robot* *Foundation Series* *Caves of Steel* What do you think? Do you want books from the above?
Me: I like science fiction. I think Asimov is awful.
System: You could try: *H. G. Wells* *Jules Verne* *George Orwell* *Aldous Huxley* What do you think? Do you want books from the above?

Table 1: Multiple Sentence Inputs

Wodehouse. Here, we first identify humor as a keyword from the first sentence; since the structure of the ontology classifies authors under genres, in this case we return a list of authors classified under *humor* to the user. The next statement is assumed to be related to the first; it specifies that the user does not like the author *Wodehouse*. This keyword is added to the reject list; when the list is specified to the user, we search for and omit *Wodehouse* from the list (Table 1).

Through the dialogue for this user, the keyword *Wodehouse* stays in the dialogue manager's reject list, so that any future cases are also eliminated. This also helps us omit references in cases where it is not immediately applicable; for example, if the user input was *I like books with humor. I don't like the Jeeves stories.*, we would add that to the reject list, and if the user chose to look at books written by *Wodehouse* in a future dialogue, we omit the books which are about *Jeeves*. Some other pragmatic instances for elimination can also be considered (*I have already read the Jeeves stories*).

Similarly, it is also possible to answer questions which are more descriptive in style. Consider, *I am trying to find a poem I read back in school. There's a man who shoots a bird or something and things start to go wrong. I think he was a sailor.* We get a set of related keywords from each utterance: *poem-school; man-shoots-bird; sailor etc.*

4.4 Identifying Relationships in the Knowledge Base

Given a set of related keywords, the system searches through the knowledge base and matches them to the entities it finds there. In order to make the search efficient and the results relevant, the search is prioritised. For example, the *x-of* relationships are searched for matches first; next the search moves through the entities through the hierarchy in order, ending at tags.

So, in our *"I want books by Dan Brown"*, we found that *"books"* and *"Dan Brown"* are the related keywords; the system finds *books* is an *x-of* relationship as part of the knowledge base, and *Dan Brown* is an entity under *author* which has an *x-of* relationship with *books*. The system therefore returns a list of books under the author *Dan Brown*. In the case of *'What books on psychology are available?'*, related keywords *"books"* and *"psychology"* are identified as an *x-of* and an entity under either *genre* or *tag* respectively, and the system similarly returns books classified or tagged as *psychology*.

In case of multiple sentences, the keyword relationships are identified individually from each sentence and then collated. This helps us retrieve a list (*I like books with humor"*) and then eliminate unwanted results (*I don't like Wodehouse*); or alternatively narrow down possibilities. Consider *"I am trying to find a poem I read back in school. There's a man who shoots a bird or something and things start to go wrong. I think he was a sailor."*. We identify *poem* as a *genre* and *school* as a *tag* or perhaps a *title*; once we add *man/shoot/bird/sailor* we continue to narrow the possibilities down and perhaps finally find all of them as *tags*. The system then returns the relevant title *(Rime of the Ancient Mariner)*.

4.5 Dialogue Flow

The hierarchy of the knowledge base guides the flow of the dialogue. The dialogue manager traverses the hierarchy and locates itself on one of the nodes, and uses the meta information of the node (genre/author, etc) to frame its replies or ask further questions. If user input is unclear and the manager cannot locate itself in the knowledge base, the system asks the user to rephrase.

Typically a user's input is processed, the keywords and the rejects identified, and the relevant results are displayed. The user may then choose to move up or down the hierarchy, or may change the topic entirely. This is identified by the break in the chain of the path being followed that is, if the user is not simply moving up or down the path but breaks the hierarchy chain such that the dialogue manager must relocate itself in the knowledge base, then the topic (or user goal) is assumed to have changed.

Consider a user who asks for some *poetry*. The user is presented with a list of authors, say *Yeats, Shelley, Wordsworth* and *Coleridge*. The chain is now from *genre* to *authors*. Next, the user may choose one of these authors, say *Wordsworth*. Then a list of poems written by *Wordsworth* is presented to the user. Now the chain goes from *genre* to *authors* to *poems*, and so on.

Similarly, a move up the hierarchy is also possible. A user may say that he wants books like, say, *The Time Machine*. The system may then present the user with a list of books written by the same author, i.e., *H. G. Wells*. The chain has now formed from *books* to *authors*. The user may express interest in other authors who write similar books; the chain then moves up to *genre* and the user is presented with a list of *authors* who write *Science Fiction*, and so on.

4.6 Topic Change

The user may explicitly jump around the hierarchy, making the dialogue mixed-initiative. For example, the user may choose to go back to the list of authors under a genre after being presented by a list of books (*go back to fantasy*) or the user may skip a level by directly specifying the book he's interested in when presented with the authors (*do you have The Adventures of Tom Sawyer by Mark Twain?*) (see Table 7 in Example Dialogues). At any point, if the user does not move up or down from the current location in the hierarchical structure, then the chain is broken. A change in user goal is assumed to have occurred, and the dialogue manager relocates the current reference node in the hierarchy. The chain is restarted from the location which is specified by the keywords obtained from the user's input.

4.7 Ambiguities and Clarification Dialogue

In order to resolve certain inherent ambiguities, the system engages the user in a simple clarification dialogue. For example, if the user asks for *books with animals in them*, the keyword relationships set up may match with books with *animals* in the title *('Animal Farm')* or generally books tagged with *animals (Harry Potter, Black Beauty, Animal Farm)*. In this case, the system will ask the user by presenting each option in order of priority until the user accepts the results.

Queries such as *books by Dan Brown and Ayn Rand* can also be considered ambiguous in the sense that the user may be looking for an intersection (books written by *both Dan Brown and Ayn Rand* or a union *books by Dan Brown and books by Ayn Rand* (which is a case of ellipsis). In such cases, the system asks the user if s/he wants the results of the intersection, and if rejected, displays results of the union (see Table 11 in Example Dialogues). In case the system does not find any results for intersection, the system specifies so and automatically presents results of the union.

The system only looks for negative indication in clarification *(no, I don't think so, etc.)* in order to be able to accept a wider range of positive indications *(yes, alright, okay, sure etc.)*.

5 Domain Independence

The system utilises the structure of the data to recognise goals and paths; there is no domain knowledge involved in the dialogue manager itself, or in fact any other module. The knowledge base is therefore replaceable, and the system is able to work in the same way for a different domain. In order to demonstrate this, examples are provided in Tables 2, 3, and 4 (additionally, Tables 9 and 10 in Example Dialogs) to show that the system works just as well with a similarly structured knowledge base for a different domain. Here, in the movies domain the *x-of* relationships are in a hierarchy consisting of *genre, director, title, actors, plot points*, etc. For the restaurant domain the *x-of* relationships are *location, cuisines, name, menu, description*, etc.

6 Evaluation

6.1 Accuracy

The system was tested on a set of questions for the books domain obtained through an online survey. A human evaluation of the system's responses was conducted based on relevance, appropriateness and correctness. The evaluation is based on whether the intended keywords and any associated

Me: I like science fiction.
System: I found these under Science Fiction: *H. G. Wells* *Jules Verne* *Isaac Asimov* *George Orwell* *Aldous Huxley* What do you think? Do you want books from the above?
Me: I will try Isaac Asimov's books. **System:** I found these under Isaac Asimov: *I, Robot* *Caves of Steel* *Foundation Series*
Me: What is in the Foundation Series?
System: I've found a book series: *1. Prelude to Foundation* *2. Forward the Foundation* *3. Foundation* *4. Foundation and Empire* *5. Second Foundation* *6. Foundation's Edge* *7. Foundation and Earth*

Table 2: Books: Moving Through the Hierarchy

System: How can I help you?
Me: I like science fiction.
System: I found these under science fiction: *Steven Spielberg* What do you think? Do you want movies from the above?
Me: I will try Steven Spielberg movies. **System:** I found these under Spielberg: *Indiana Jones* *Schindler's List* *Jurassic Park* *Catch Me If You Can* *Saving Private Ryan*
Me: Please tell me about Indiana Jones.
System: I've found a movie series: *1. Raiders of the Lost Ark* *2. Temple of Doom* *3. Last Crusade* *4. Kingdom of the Crystal Skull*

Table 3: Movies: Moving Through the Hierarchy

constraints are correctly identified. A result is considered to be correct if the relevant or expected output is returned. An overall accuracy of 88% was achieved over 50 questions (Table 5).

A second evaluation was carried out to test the dialogue system qualitatively by means of user responses to questions in a Likert-scaled questionnaire; 8 participants were asked to interact with the system (over about 8-10 turns) and rate it on a scale of 5 on the basis of relevance (whether they thought the system understood their goals), correctness (compare expected response with actual system response) and an overall rating (navigation, task completion, etc). A Wizard-of-Oz benchmark was also collected. The results are in Table 6.

6.2 Error Analysis

There are many errors introduced due to ambiguous structures, grammatically unsound instances, etc. Parsing errors, like wrong identification of root, incorrect argument assignment, etc. also introduce significant error as the system is unable to identify the right keywords.

Parsing Issues: There is a prepositional phrase attachment problem - consider the following queries:
(a) *Do you have Catcher in the Rye?*
(a) *Do you have Of Mice and Men in the library?*
(b) *Do you have Body in the Library?*

In (a), the prepositional phrase *in the Rye* is part of the title and important information; in (b) *in the library* is irrelevant information. It may also not be possible to ignore this phrase on the basis of the domain, as we can see from (c) where the title of the book itself has *in the Library* in it. In such cases, we may end up including irrelevant keywords and retrieving wrong information. Similar problems also occur when the book titles are phrases or sentences themselves (*Do you have Where the Wild Things Are?*) which are considered as part of the sentence by the tagger and assigned incorrect roots or copula tags.

Pragmatics: Consider *'Which is the most popular dish in RESTAURANT-2?'*. Converting *'the most popular'* to mean that we need to group by dish name for the restaurant, then look at the ratings, frequency of orders or other relevant information, is hard to do even if such information is available in the knowledge base, and such an inference includes information not present in the query. Also, the thresholds of such subjective opinions may differ.

Recognizing the User's Goal: Consider the example, *What other books are there have that*

System: How can I help you?
Me: I like chinese food.
System: I found these under chinese food: *RESTAURANT-16* *RESTAURANT-24* What do you think? Do you want menus from the above?
Me: I will try RESTAURANT-24. **System:** I found these under RESTAURANT-24: *Hakka Noodles* *Chicken Fried Rice* *Chilli Mushroom* *Chicken 65*

Table 4: Restaurants: Moving through the Hierarchy

Type of Input	Accuracy
Single line	88.09%
Relative clauses	66.66%
Multiple lines	100%

Table 5: Survey Query Evaluation

were written by the author of The Old Man and the Sea?. Here, we need to first identify the *author* of the *The Old Man and the Sea* as Hemingway, and then retrieve books written by Hemingway. This retrieval may be complex and hard to identify.

7 Future Work

Pre-processing problems involving spelling and grammatical errors, synonyms, missing arguments, abbreviations, etc. need to be handled in order to make the system practically useful. Also, the words must be lemmatized and replaced with a mapped, representative synonym that is part of the knowledge base. A chunker or an NER can be used to identify complete constraints with functions words which the parser might separate (like *'Harry Potter and the Goblet of Fire'*). Ellipsis and anaphora issues need to be identified in depth.

A more robustly domain-independent system which can work despite significant changes in the knowledge base structure must be developed. For example, the system should be capable of handling data having complex networks and interconnections among the entities (instead of a simple hierarchy as we have used).

Instead of a simple dictionary, a better method for analyzing the positively and negatively associated words expressed in queries can be found, instead of a simple dictionary. Keeping track of both positive and negatively associated keywords could prove helpful for providing suggestions.

Category	Woz Benchmark	Avg. Rating out of 5
Relevance	4.625	4
Correctness	4.5	3.5
Overall	4.25	3.5

Table 6: Subjective Evaluation

A corpus with a wide variation in the pattern of questions, along with questions in statement form must be analyzed and considered, in order to make the system more accurate for generic queries. The usefulness of the information gathered through this method and the ease of mapping to a data representation must also be studied. More rigorous testing needs to be done for the descriptive questions.

The possibility of increasing the accuracy of keyword identification by using statistical machine learning methods also needs to be explored.

References

Akshar Bharati, Ashok Gupta, and Rajeev Sangal. 1995. Parsing with nesting constraints. In *Proc of 3rd NLP Pacific Rim Symposium*, pages 4–6.

Timothy W Bickmore, Daniel Schulman, and Candace L Sidner. 2011. A reusable framework for health counseling dialogue systems based on a behavioral medicine ontology. *Journal of biomedical informatics*, 44(2):183–197.

Dan Bohus and Alexander I Rudnicky. 2003. Ravenclaw: Dialog management using hierarchical task decomposition and an expectation agenda. *Eighth European Conference on Speech Communication and Technology*.

Myroslava O Dzikovska, James F Allen, and Mary D Swift. 2003. Integrating linguistic and domain knowledge for spoken dialogue systems in multiple domains. In *Proc. of IJCAI-03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

Annika Flycht-Eriksson. 2004. Design and use of ontologies in information-providing dialogue systems. *Dissertation, Linkping Studies in Science and Technology, Thesis n. 874, School of Engineering, Linkping University*.

Prathyusha Jwalapuram and Radhika Mamidi. 2017. Keyword and constraint identification for question answering. In *15th International Conference of the*

Pacific Association for Computational Linguistics, Yangon, Myanmar, in print.

Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, and Gary Geunbae Lee. 2009. Example-based dialog modeling for practical multi-domain dialog system. *Speech Communication*, 51(5):466–484.

Laurent Mazuel and Nicolas Sabouret. 2006. Generic natural language command interpretation in ontology-based dialogue systems. In *Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology*, pages 347–350. IEEE Computer Society.

David Milward and Martin Beveridge. 2003. Ontology-based dialogue systems. In *Proc. 3rd Workshop on Knowledge and reasoning in practical dialogue systems (IJCAI03)*, pages 9–18.

A Example Dialogues

System: What would you like me to look for?
Me: I want to try fantasy fiction.
System: I've found Fantasy. I found these about/under Fantasy : *J. K. Rowling* *Christopher Paolini* *George R. R. Martin* *Jonathan Stroud*
Me: Do you have books by Rice?
System: I've found Anne Rice. I found these about/under Anne Rice : *The Vampire Chronicles* What would you like me to look for?
Me: Go back to fantasy.
System: I found these about/under Fantasy : *J. K. Rowling* *Christopher Paolini* *George R. R. Martin* *Jonathan Stroud*

Table 7: Books: Topic Change

System: What would you like me to look for?
Me: I like books which are about vampires.
System: I've found vampires. Would you like Dracula?
Me: I would prefer books which have some mystery in them.
System: I've found Mystery, Thriller, Suspense in the database.

Table 8: Books: Relative Clauses

Me: I like Indian food. Like Chicken Tikka.
System: I found these under Indian: *RESTAURANT-13* *RESTAURANT-22* *RESTAURANT-26* I found these for Chicken Tikka: *RESTAURANT-22* *RESTAURANT-26*
Me: I like Indian food. But I don't like RESTAURANT-22 or RESTAURANT-26.
System: I think these restaurants are relevant: *RESTAURANT-13*

Table 9: Restaurants: Multiple Sentence Input and Rejection

Me: I like Tom Hanks.
System: I think these movies are relevant: *Catch Me If You Can* *Saving Private Ryan*
Me: Recommend some Matt Damon movies. **System:** I think these movies are relevant: *Saving Private Ryan*
Me: I like Hanks. I don't like Damon.
System: I think these movies are relevant: *Catch Me If You Can*

Table 10: Movies: Multiple Sentence Input and Rejection

System: What would you like me to look for?
Me: I want Neil Gaiman and Terry Pratchett books.
System: Do you want an intersection of Neil Gaiman and Terry Pratchett?
Me: Yes.
System: Would you like Good Omens?
System: What would you like me to look for?
Me: I like Jules Verne and Isaac Asimov.
System: Do you want an intersection of Jules Verne and Isaac Asimov?
Me: No.
System: I found these under Jules Verne *Twenty Thousand Leagues Under the Sea* *Around the World in Eighty Days* *Journey to the Center of the Earth* I found these under Isaac Asimov: *I, Robot* *Caves of Steel* *Foundation Series* What do you think? Do you want books from the above?

Table 11: Books: Clarification Dialogue

Document Level Novelty Detection : Textual Entailment Lends a Helping Hand

Tanik Saikh, Tirthankar Ghosal, Asif Ekbal, Pushpak Bhattacharyya
Department of Computer Science and Engineering, Indian Institute of Technology Patna
{tanik.srf17,tirthankar.pcs16,asif,pb}@iitp.ac.in

Abstract

In this paper we present a way of detecting novelty of a document with respect to the relevant source documents with the aid of methods used in detecting Textual Entailment (TE). The proposed TE system is based on supervised machine learning approach that makes use of different similarity metrics. The TE system is further interpreted to detect the novelty of an incoming document with respect to some source document(s) already seen by the system. We design a corpus to support this foundation of novelty at the document level and coin it as the *Document Level Novelty Detection (DLND)* corpus. We employ standard supervised classification algorithms such as Support Vector Machine (SVM), Multilayer Perceptron (MLP) and Random Forest (RF) and investigate their performance on DLND. Evaluation results show the accuracies of 78.78%, 77.27% and 74.24% for SVM, MLP and RF, respectively on DLND. To establish the efficacy of our methods we evaluate our model on the benchmark datasets released in the shared task of *Recognizing Textual Entailment - 6 (RTE-6)* and *Recognizing Textual Entailment - 7 (RTE-7)*. Experiments show the accuracies of 94.91% and 96.72% on RTE-6 and RTE-7 dataset, respectively.

1 Introduction

Novelty detection from texts is an age-old problem in text mining and have found significance in various applications of Natural Language Processing (NLP) such as Text Summarization (Bysani, 2010). Novelty detection from texts implies figuring out new information from a given piece of text and subsequently arriving to the judgment that whether a given piece of text could be termed as novel or not. The decision should always be with respect to some relevant pieces of source texts. The problem of novelty detection has been studied via various NLP and machine learning (ML) paradigms ranging from classification to clustering. On the other hand TE is a NLP problem which is defined as a directional relationship between the two text fragments, termed as *Text (T)* and *Hypothesis (H)*. It is said that:

***T** entails **H** if, typically, a human reading **T** would infer that **H** is most likely to be true* (Dagan et al., 2006)

i.e. to judge that whether *H* could be inferred from *T*. This inference is not only based on understanding of T but also on some prior domain knowledge. Novelty detection finds it's relevance with TE in the sense that, a certain hypothesis *H* entailed from a certain piece of source text *T* could be considered as non-novel with respect to *T* if a human reading the hypothesis *H* after reading *T* would find redundant information in *H*. Whereas if *H* is not entailed from *T* then a human reading *H* after *T* would find new piece of information in *H* and hence *H* could be considered as novel with respect to *T*. The basis of our work also proceeds with this intuition and is grounded with the very basic relationships of textual entailment with textual similarity. Textual similarity is bi-directional relationship between two text fragments whereas textual entailment is an uni-directional relationship between the hypothesis and source text where the former could be derived from the latter but not the reverse. Similarity, it can be manifested in a scale that

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 131–140

ranges from semantic equivalence to complete unrelatedness, whereas TE can be either *Yes* or *No*. The implication of novelty with TE was first attempted in the TAC RTE-6 Novelty Detection Subtask (Bentivogli, 2010) and also being carried out in RTE-7 (Bentivogli, 2011). In these tracks also they defined those piece of Hypotheses as *Novel* which are *Not Entailed* by Texts. On the basis of this intuition we carry out the experiments described henceforth. These tasks were rendered at the sentence-level and they established this view of TE as an opposite characteristic to novelty. In this work we take forward this view to investigate novelty detection at the document level via TE with emphasis to textual similarity measures. The contributions of the present work could be enumerated as follows:

- Investigating the role of TE to detect novelty of a document.
- Creating our own benchmark corpora for novelty detection at the document level.

1.1 Motivation

The motivation behind the current work stemmed from the following:

- Exponential dump of redundant information across the web which hinders user quest of new meaningful pieces of information.
- Explore the implication of TE to detect novelty at the document level.

We make use of lexical level similarity features to build the TE system. The studies (Saikh et al., 2015; Saikh et al., 2016) showed that the use of similarity measures such as *Cosine Similarity, Jaccard, Dice, Overlap etc.* as features can effectively be used in taking entailment decision between a pair of texts (RTEs datasets) and these were also used in detecting paraphrase relations between a pair of texts written in Indian languages (Tamil, Malayalam, Hindi and Punjabi) as in FIRE-2016 shared task, namely *Shared Task on Detecting Paraphrases in Indian languages (DPIL)*. This straightforward relationship between textual similarity and TE encouraged us to explore various similarity measures to detect entailment at the document level. Entailment criteria lead us to investigate the novelty of the target text with respect to a set of source text(s). Our understanding and survey reveal that in spite of having great potential in various applications, novelty detection at the document level did not garner required attention. Thus investigating textual similarity measures to infer document level entailment formed the very basis of our work for detection of novelty at the document level. To the best of our knowledge our approach in viewing document level novelty detection task is novel and has not been tried before. We believe that our method towards detecting novelty of a document correlating with textual entailment would provide a strong baseline and instigate further research along this line.

1.2 Related works

Research in novelty detection could be traced back to the Topic Detection and Tracking (TDT) (Wayne, 1997) evaluation campaigns where the concern was First Story Detection (FSD) or to detect new events with respect to online news streams, notable being the UMass approach (Allan et al., 2000). The task gained popularity in the tracks of Text Retrieval Conferences (TREC) of the year of 2002, 2003 and 2004 (Voorhees, 2002; Voorhees, 2003; Clarke et al., 2004) although the focus was at sentence level novelty detection. Some interesting works in TREC were based on the sets of terms (Zhang et al., 2003a; Zhang et al., 2003b), term translations (Collins-Thompson et al., 2002), Principal Component Analysis (PCA) vectors (Ru et al., 2004), SVM classification (Tomiyama et al., 2004) etc. Similar works relied on named entities (Gabrilovich et al., 2004; Li and Croft, 2005; Zhang and Tsai, 2009), language models (Zhang et al., 2002; Allan et al., 2003), contexts (Schiffman and McKeown, 2005) etc. At the document level, (Karkali et al., 2013) computed novelty score based on the inverse document frequency scoring function. More recently (Dasgupta and Dey, 2016) conducted experiments with information entropy measure to calculate innovativeness of a document. Novelty detection with the help of TE was first introduced as a subtask of RTE-6 (Bentivogli, 2010) challenge organized by Text Analysis Conference in the year of 2010. Several participants took part in this shared task and reported various interesting results which opened a new avenue of determining novelty with the help of TE. The best result was obtained by (Houping Jia and Xiao, 2010) with an

F-Score of 82.91%. The authors made use of Syntactic method (MINIPAR parser relationship) and semantic knowledge (Wordnet, Verb Ocean and LingPipe) to achieve the accuracy. The novelty detection subtask was again organized as a part of RTE-7 (Bentivogli, 2011). In this track the best F-Score of 90.95% was obtained by (Tsuchida and Ishikawa, 2011). Their machine learning based approach employed lexical level matching measures as features. Other participating system's results in this track were very promising and revealed that detecting novelty using entailment could be a good direction. We leverage this idea of TE for detecting novelty but at the document level. Due to the non-availability of a proper, dedicated document level novelty detection corpus, we create a dataset for the purpose. We use supervised machine learning algorithms : SVM (Vapnik, 1995; Chang and Lin, 2011), RF (Breiman, 2001) and MLP (Becerra R., 2013; Costa et al., 2015) on features extracted from our as well as RTE datasets. Evaluation shows encouraging performance on both the datasets as reported in Section 4.

2 Proposed Method for Novelty Detection

We propose a supervised scheme for detecting document level novelty using the features for detecting TE. The proposed method aims at developing a machine learning based TE system where different similarity measures were employed as features. The features include vector based similarity measures (i.e. cosine, Dice), set based similarity measures (i.e. Jaccard, Overlap and harmonic), lexical level similarity measures (i.e. unigram similarity with respect to novel/non-novel, unigram similarity with respect to source), entailment trigger polarity based similarity (based on negation), the length difference between text and hypothesis, the number of overlapping keywords and the number of overlapping Named Entities (NEs). Given a pair of documents (i.e. target-source) the system has to decide whether the target document can be entailed from any of the source(s). A document is treated as non-novel if it is fully entailed from any or all of the source documents. Else if there is sufficient new information in the target document which is not derived from the source(s), the document is viewed as novel. Paucity of a dedicated document level novelty detection corpus led us to create the corpus and we term the resource as the *Document Level Novelty Detection (DLND)* corpus. It consists of 202 different topics mostly taken from the *politics* and *business* domains. In each topic there exists at least one novel and non-novel documents and three source documents. Each target (novel or non-novel) document is compared with three source documents on the same topic. We calculate similarity scores between a target document and three on-topic source documents with the help of above mentioned measures. So for each target document pitched against the three source documents, we obtain three scores for each feature. Hence we rely on two methods, namely *Maximum* and *Average* to arrive upon the final measure.

1. *Maximum:* For each topic, each target document is compared with all the three source documents. This yields three scores for each similarity measure. We take the maximum of the three values with the intuition that a *non-novel* document would have a high similarity score with all or any one of the source document(s). Whereas a *novel* document would contain new information and would be lexically distant from all the three source documents. Hence even if we take the maximum of the similarity values, it would yield low score as compared to that of the *non-novel* documents. Let us consider there is a *novel/non-novel* target document d_t which is to be compared with three source documents d_{s1}, d_{s2} and d_{s3}. For each feature, we thus compute three scores sc_1, sc_2 and sc_3. We take the maximum of these three scores as the feature value for the respective feature.

2. *Averaging:* In this approach we take the average of the three scores obtained against the three source documents. This we do assuming that reference information is distributed in the source documents. So for a target document d_t with three source documents, d_{s1}, d_{s2} and d_{s3}, we hence obtain three scores (for each feature) sc_1, sc_2 and sc_3. We take the average of these three scores as feature value for the respective features.

For each instance we generate the feature vector consisting of all the features as mentioned above. We assign the class label as *Not Entailed*, when

we compare with a novel document and as *Entailed* when the comparison is performed with a non-novel document. We assume a piece of text as *Novel* which is *Not Entailed* with respect to the set of repositories (source documents). Such relation between novelty and TE was established in the subtask, namely novelty detection using textual entailment in (Bentivogli, 2010; Bentivogli, 2011). We develop models using three popular supervised machine learning algorithms, namely SVM with linear Kernel (Vapnik, 1995; Chang and Lin, 2011), MLP (Becerra R., 2013; Costa et al., 2015), and RF (Breiman, 2001). SVM is known to be one of the very promising classifiers for binary classification. MLP makes use of back-propagation to classify instances and random forest combines the output of multiple decision tree which is a tree based classifier. We make use of Weka [1] implementation of these classification algorithms.

2.1 Features used for Novelty Detection

Features play very crucial role in any machine learning assisted experiment. Hence, use of proper features for solving the problem is an important part of such a particular system. We use the following set of features for training and testing of classifiers:

1. *Cosine Similarity:* Cosine similarity (Nguyen H.V., 2011) is a vector based similarity metric. It calculates similarity between the two vectors of A and B by the following formula. This is a well known similarity metric and perhaps the most widely used one.

$$Cos\theta = A.B/||A||.||B|| \quad (1)$$

   where, A and B are two vector representations of two texts. The similarity score lies between 0 to -1, where, -1 indicates exactly opposite, 1 indicates exactly same, and 0 indicates the independence. It is to be assumed that higher the similarity score obtained more is the chance that the pair of text snippets become textually entailed, so it could be a good predictor of TE.

2. *Jaccard Similarity:* Jaccard similarity (Jaccard, 1901) is a set based similarity metric. It is defined as follows:

$$Jaccard(A, B) = |A \cap B|/|A \cup B| \quad (2)$$

   where A and B represent two sets of documents. A similar pair is expected to share more words and hence the entailment relation holds (Almarwani and Diab, 2017). Following this intuition we make use of set based similarity metric in our work. This is very well established similarity metric and measure the similarity between the two finite sets.

3. *Dice Similarity:* Dice Similarity (Dice., 1945) is also a vector based similarity metric. It's value lies within the range of 0 to 1. It can be calculated using the following formula.

$$Dice(A, B) = 2|A \cap B|/(|A| + |B|) \quad (3)$$

   Here, A and B represent the first and second set of documents, respectively. The mathematical derivation of this measure is same as the derivation of F-measure, where precision and recall both are taken into account. So this measure also captures both precision and recall.

4. *Overlap:* Overlap (Jayapal, 2012) is another set based similarity metric, where a discourse can be represented by a set. Elements of the set are words. It's value lies between 0 to 1. It can be calculated as per the following equation:

$$Overlap(A, B) = |A \cap B|/min(|A|, |B|) \quad (4)$$

   Here, A and B correspond to the Bag-of-Words (BoW) representation of two comparing documents.

5. *Harmonic:* Harmonic (Joshi et al., 2007) is a set based similarity metric. It can measure the similarity between two pairs of documents by the following equation

$$Harmonic(A, B) = |A \cap B|(|A + |B|)/2.|A|.|B| \quad (5)$$

   Here A and B representing two comparing documents in terms of set.

6. *Unigram similarity with respect to target document:* Here we measure the similarity between two documents by calculating the number of common unigrams between a pair of comparing documents normalized by the number of unigrams present in novel/non-novel (target) document to which the comparison is being performed. This can be illustrated by the following equation, where *nuc:*

[1] http://www.cs.waikato.ac.nz/ml/weka/

Number of common unigrams in two documents and *nut: Number of unigrams in the target document.*

$$US_t = \frac{nuc}{nut}$$

More is the overlapping of unigrams between the two documents higher is the chance of entailment between these.

7. *Unigram Similarity with respect to source document:* Unigram similarity with respect to source document is computed following the same way as the previous approach, *except* the normalization is done by the number of unigrams present in the source document. This can be represented by the following formula, where *nuc: Number of unigrams common between two documents* and *nus: Number of unigrams in source document*

$$U.S_s = \frac{nuc}{nus}$$

8. *Length difference:* The length difference between the two comparing documents is used as a feature. Our analysis to the datasets released as part of RTE-1 to RTE-5 show that length of *"Text (T)" -the entailing text* is always larger than the length of *"Hypothesis (H)" - the entailed hypothesis* as shown in Table 1, where, THP : number of T-H pairs, ATL: average text length in words and AHL : average hypothesis length in words for the development and the test set belonging to each dataset. These statistics, therefore, shows that the length difference can be used as a feature in the experiment.

Datasets	Development set			Test Set		
	THP	ATL	AHL	THP	ATL	THP
RTE-1	567	23	9	800	25	10
RTE-2	800	26	9	800	27	8
RTE-3	800	34	8	800	29	7
RTE-4	0	0	0	1000	39	7
RTE-5	600	97	7	600	96	7

Table 1: Statistics of the RTEs datasets

9. *Number of overlapping keywords:* The meaning of a textual document is often represented by a set of keywords. We extract the keywords present in each source and target document. we make use of Rapid Automatic Keyword Extractor (RAKE) [2] (Rose S. and W., 2010) for this purpose. We count the number of overlapping keywords between the two (source and target) comparing documents. This count is set as the feature value in our experiment.

10. *Number of overlapping Named Entities (NEs):* Named entities (NEs) provide important evidence in taking the entailment decision between a pair of texts. We use Stanford NER[3] for recognizing the NEs. We extract NEs present in novel, non-novel and source document and find the number of overlapping NEs between the two (source and target) comparing documents. We use this count as the feature value in our experiment.

11. *Polarity feature:* Most of the features used in our work are based on lexical matching. Presence of negation might cause a problem in the entailment decision if we rely solely on the lexical matches. As an example, let us consider the following two sentences: *T: Puja lives in Delhi.* and *H: Puja does not live in Delhi*, If we compare these two sentences using lexical matching it will produce a considerably high similarity score. Thus the system will decide these as textually entailed, but actually they are not so. In order to handle this situation we define the feature as following. A document might contain more than one negation words. In order to handle negation at the document level we make use of stanford NER tagger and RAKE key phrase extractor to identify NEs and keywords present in a particular document. In every sentence in a document we search for the keyword or NE. If any of these or both are present in a sentence, we pick up those sentences. We count the number of negation words like *"no/not"* present in those sentences. We take those count as the feature value. This is a very trivial approach and needs further investigation.

3 Dataset Description

We evaluate the efficacy of our approach on the RTE-6 and RTE-7 datasets for novelty detection subtask. It is to be noted that these two datasets were created aiming sentence-level novelty detection. However in the present work we focus on detecting document level novelty. To investigate

[2]https://github.com/aneesha/RAKE

[3]https://nlp.stanford.edu/software/CRF-NER.html

the implication of our methods for detecting novelty of a document we create the *Document Level Novelty Detection (DLND)* corpus.

3.1 Benchmark Datasets (RTE-6/7)

The novelty detection subtask was organized in conjunction with the main tasks of RTE-6 (Bentivogli, 2010) and RTE-7 (Bentivogli, 2011) tracks. In these tracks, organizers released a benchmark dataset for novelty detection using TE. We make use of this corpus to evaluate our system. In RTE-6 the novelty detection dataset consists both development and test sets. Each set contains 10 different topics. Statistics of development and test sets are shown in Table 2. There exists multiple texts for each hypothesis in both development and test datasets. The entailment decisions are either **Yes, i.e Non-novel** or **No i.e Novel** for each hypothesis and text pair.

		Development Set	Test Set
RTE-6	Topics	10	10
	Hypotheses	183	199
RTE-7	Topics	10	10
	Hypotheses	284	269

Table 2: RTE-6 and RTE-7 Novelty Subtask Dataset Statistics

3.2 DLND Corpus

We prepare the *Document Level Novelty Detection (DLND)* corpus by **unbiased** topic-wise crawling of newspaper articles belonging mostly to politics and business genre for a period of five months (from November 2016 - March 2017). The objective was to investigate, that for a given set of on-topic relevant documents already seen/read by the user, what is the novelty of an incoming on-topic document to him/her? We follow the heuristics that, on a given date, different newspapers would report similar contents regarding a specific event, and hence be content-wise non-novel to a reader once s/he had already read one of them. Reporting on subsequent dates on the same event would contain some new information, hence could be considered as novel. For this we keep three on-event reporting by different agencies as the **Source** documents usually chosen from the initial dates of reporting. Having read the source documents we ask the annotators to annotate the on-event other crawled documents from different dates as **non-novel** or **novel** with respect to the source collection based on the information coverage and human judgment. The final structure of DLND corpus looks like as shown in Figure 1. Three annotators with post-graduate level of knowledge in English were employed to use their expertise for labeling an incoming target document as *novel* if the target document has minimum semantic/lexical overlap with the source documents. A certain target document was labeled as *non-novel* if there was maximum lexical/semantic overlap with the source documents. We left out the indecisive cases for our experiments. We found the inter-rater agreement to be **0.82** in terms of **Kappa co-efficient** (Cohen, 1960) which is considered to be good as per (Landis and Koch, 1977). Intuitively, we perceive

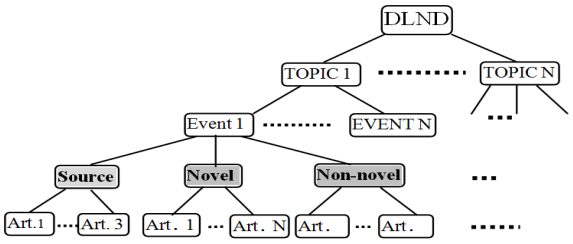


Figure 1: DLND corpus structure

the source collection of each event as the memory of the reader whereas novel and non-novel are the test instances against the knowledge of the reader. The datasets consists of 202 different topics. For each topic there exist at least one novel/non-novel document and three source documents. We partition the whole corpus into train and test sets following the ratio of 7:3. Statistics of the datasets for training and test set in terms of average document length in three categories, namely Novel, Non-Novel and Source documents are shown in Table 3.

	Novel	Non-Novel	Source
Training Set	3057	2337	2908
Test Set	1310	1001	1246

Table 3: Statistics of the DLND Datasets

4 Experiments, Results and Discussion

In this section we discuss the pre-processing done on the datasets, results obtained through experimentations and thereby analyze the errors. As the documents were collected from the various web sources, these were not well structured. We pre-processed the documents by removing white spaces.

4.1 Experiments

We calculate similarity scores between a target (novel/non-novel) and source document using various similarity measures, and use these as features in our classifiers. These scores are used to generate the feature vectors for classifier's training and/or testing. As already mentioned we used RF, MLP and SVM as our classification algorithms. These models are used to assign a class label (*Entailed or Not Entailed*) to each instance in the test set. These predicted classes are compared to the gold label to compute the final results.

Novelty and TE are highly co-related. In the context of similarity, a target document is said to be novel with respect to a collection of source document(s) if it has very less similarity to the sources. Otherwise, it is termed as novel. On the other hand similarity and TE are directly proportional if we keep aside the presence of negations in the comparing texts. TE between two texts can be judged by measuring the similarity between those two particular texts. We can conclude that novelty and TE are opposed to each other. Entailment can be a way of judging the non-novelty of a document. We report the results on test set of different classifiers in Table 4. Results reported in

	Accuracy (Percentage)	
Classifiers	**Maximum**	**Averaging**
SVM (Best Performing Classifier)	**78.78**	**78.55**
MLP	77.27	75.61
RF	74.24	69.73

Table 4: Results on DLND test datasets

Table 4 demonstrate that SVM in both the cases performs best amongst all. This is not unexpected keeping in mind the success of SVM in solving a wide range of text classification problems with features which are overlapping in nature. MLP makes use of back-propagation technique to classify instances. In our setting we use 5 layers that might have caused better accuracy. Random Forest also seems to suit well to our task.

4.2 Results on benchmark datasets

We evaluate our model on the benchmark datasets of RTE-6 and RTE-7 for novelty detection. The task was to detect those hypotheses which are novel (not-entailed) with respect to the corpus. We show the results in Table 5, where P: Precision, R: Recall and F1: F-score. We also compare the performance with the best systems reported in RTE-6 (Houping Jia and Xiao, 2010) and also in RTE-7(Tsuchida and Ishikawa, 2011). The best result obtained in RTE-6 novelty detection subtask is with the F-score of 82.91% by (Houping Jia and Xiao, 2010). Syntactic (output of MINIPAR parser, nodes matching texts and hypotheses) and semantic (WordNet, Verb Ocean, and LingPipe) matching between texts and hypotheses were employed for that purpose. An F-score of 90.95% was obtained as the best score by (Tsuchida and Ishikawa, 2011) on RTE-7 novelty detection dataset, with mostly lexical matching features in a machine learning framework. As is evident, our proposed system successfully outperforms those state-of-the-art techniques of RTE-6 and RTE-7 by a significant margin.

		P(%)	R(%)	F1(%)
RTE-6	(Houping Jia and Xiao, 2010)	72.39	97	82.91
	Proposed Method	**95.74**	**99.08**	**96.86**
RTE-7	(Tsuchida and Ishikawa, 2011)	86.92	95.38	90.95
	Proposed Method	**96.97**	**99.73**	**98.33**

Table 5: comparison of results obtained with the best system's results on RTE-6 and RTE-7

4.3 Sensitivity Analysis of Features

In order to illustrate the contribution of each feature to our predicting class, we perform an ablation study. Table 6 below reports the accuracy figures on training set (based on 10-fold cross validation) by removing one feature after another, where the acronyms *U.S.N, U.S.S, L.D, Keyword and NE stands for Unigram similarity with respect to target (Novel/Non-Novel) document, Unigram similarity with respect to source, Length Difference, number of overlapping keywords and number of overlapping Named Entities respectively*. Table 6 shows that 'unigram similarity with respect to target document', # of keywords match, # of NE match, and Cosine similarity are the most contributing features to our experiments.

Feature Removed	Accuracy (%)
None	85.38
Cosine Similarity	84.85
Jaccard Similarity	85.10
Dice	85.17
Overlapping	85.13
Harmonic	84.85
U.S.N	83.60
U.S.S	84.06
L.D	85.03
Keyword	82.70
NE	83.12
Polarity	84.96

Table 6: Feature sensitivity analysis

4.4 Error Analysis

Below we analyze the output of the system and summarize the causes of the errors committed.

1. In our current work we assumed that more the similarity at the lexical level, higher is the chance that the document pair is entailed to each other. The intuition behind this lexical matching based experiment was grounded with a very basic assumption that more the overlapping tokens between two comparing documents higher is the chance of holding TE relation between that pair of text snippets. Although this assumption works up to a certain extent, but fails when semantics is to be considered.

2. Presence of negation words often creates problem in entailment decision. To overcome this we make use of polarity based feature (i.e presence/absence of negation words). This intuition works well for the single occurrence of negation word, but as we deal with documents there might be multiple negation words in a particular document. Dealing with multiple occurrences of negation words at the document level is very challenging. We will investigate this in more details in the future.

3. Although the proposed system considers the *NEs and keywords*, but it does not take *Multiword Expressions (MWEs)* into account. Dealing with multi-word expressions may be useful in taking entailment decision.

4. One of the major drawbacks of this system is the sparsity problem. The system represents a text with lexical-level sparse vectors. So, there might be some instances (having different vocabulary) for which similarity measure can produce zero score, even though they are highly entailed.

4.5 Comparisons with the state-of-the-art

In order to compare our method with state-of-the-art systems we evaluate a recent method proposed in (Dasgupta and Dey, 2016) on our DLND corpus. This particular entropy-based approach produced novelty score (NS) of a document **d** with respect to a collection **c**. We adapt the respective threshold criteria and infer that documents with novelty score above *(average+standard deviation)* are *Novel* and that with novelty score below *(average-standard deviation)* are *Non-Novel*. We left out the remaining *average novelty* class cases.

System	Accuracy (%)	F_1 (%)
(Dasgupta and Dey, 2016)	67.94	70.34
Proposed Approach (Maximum-SVM)	**78.78**	**93.49**

Table 7: Comparison with the state-of-art

From Table 7 we could see that our proposed *Maximum* method based on SVM classifier performs better compared to the approach of (Dasgupta and Dey, 2016) by a margin of almost 11 points in terms accuracy.

4.6 Tests of Significance

To analyze if the improvement obtained in our system is statistically significant over the state-of-the art, we perform *t-test* at 5% significance level. The *p-values* for F-measures produced by 20 runs of our system against the best performing systems of RTE-6 was 5.30e-85 and for RTE-7 was 1.60e-74. We also pitched our system's F-measure against that obtained by the approach of (Dasgupta and Dey, 2016) on DLND for 20 runs and the p-value was 2.27e-91. All the p-values thus are less than 0.05 and hence the improvement is statistically significant and unlikely to be observed by chance in 95% confidence interval.

5 Conclusion and Future Works

In this work we addressed the problem of detection of novelty of a document with respect to on-topic source document(s) using the concept of TE. We built an entailment model based on supervised approaches that make use of features extracted from the different lexical level similarity metrics. We also created a dedicated resource for document level novelty detection which may pave the way for further research in this topic. Our evaluation on DLND shows promising results to serve as a strong baseline for further research. Evaluation on the RTE-6 and RTE-7 datasets demonstrate the effectiveness of our approach over the existing literature methods on novelty detection. The research carried out in these experiments opens up a new avenue for detecting novelty of text at document level using textual entailment.
In future, we would like:

1. To employ deep semantic features so that the system can capture ambiguous sentences contained in a particular document.

2. To investigate semantic textual similarity to detect novelty of a document with deep learning techniques.

3. To address the sparsity problem, we intend to incorporate WordNet based similarity measures and explicit semantic analysis that will use bag-of-word representation retrieved from the Wikipedia text. Also distributional representation of words(word2vec) may prove effective to capture semantics.

4. To see the performance of the best performing systems of RTE-6 (Houping Jia and Xiao, 2010) and in RTE-7 (Tsuchida and Ishikawa, 2011) applied to our DLND dataset.

Acknowledgments

We would like to acknowledge "Elsevier Centre of Excellence for Natural Language Processing" at IIT Patna for supporting the research work furnished here in this paper.

References

James Allan, Victor Lavrenko, Daniella Malin, and Russell Swan. 2000. Detections, Bounds, and Timelines: Umass and tdt-3. In *Proceedings of topic detection and tracking workshop*, pages 167–174.

James Allan, Courtney Wade, and Alvaro Bolivar. 2003. Retrieval and Novelty Detection at the Sentence Level. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 314–321. ACM.

Nada Almarwani and Mona Diab. 2017. Arabic Textual Entailment with Word Embeddings. In *Proceedings of The Third Arabic Natural Language Processing Workshop (WANLP)*, pages 185–190, Valencia, Spain.

Garca Bermdez R.V. Velzquez L. Rodrguez R. Pino C. Becerra R., Joya G. 2013. Saccadic Points Classification Using Multilayer Perceptron and Random Forest Classifiers in EOG Recordings of Patients with Ataxia SCA2. *(eds) Advances in Computational Intelligence. IWANN. Lecture Notes in Computer Science*, 7903(3).

Magnini B. Dagan I. Dang H.T. Giampiccolo D. Bentivogli, L. 2010. The Sixth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the Text Analysis Conference (TAC 2010), November 15-16, 2010 National Institute of Standards and Technology Gaithersburg, Maryland, USA*.

Clark P. Dagan I. Dang H. T. Giampiccolo D. Bentivogli, L. 2011. The Seventh PASCAL Recognizing Textual Entailment Challenge. In *In TAC 2011 Notebook Proceedings, November 14-15, 2011, Gaithersburg, Maryland, USA*.

Leo Breiman. 2001. Random Forests. *Mach. Learn.*, 45(1):5–32.

Praveen Bysani. 2010. Detecting Novelty in the Context of Progressive Summarization. In *Proceedings of the NAACL HLT 2010 Student Research Workshop*, pages 13–18, Los Angeles, CA, June. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27.

Charles LA Clarke, Nick Craswell, and Ian Soboroff. 2004. Overview of the TREC 2004 Terabyte Track. In *TREC*, volume 4, page 74, National Institute of Standards and Technology Gaithersburg, MD.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and psychological measurement*, 20(1):37–46.

Kevyn Collins-Thompson, Paul Ogilvie, Yi Zhang, and Jamie Callan. 2002. Information Filtering, Novelty Detection, and Named-Page Finding. In *TREC*, Gaithersburg,MD.

Wanderson Costa, Leila Maria Garcia Fonseca, and Thales Sehn Körting. 2015. Classifying Grasslands and Cultivated Pastures in the Brazilian Cerrado Using Support Vector Machines, Multilayer Perceptrons and Autoencoders. In *Machine Learning and Data Mining in Pattern Recognition - 11th International Conference, MLDM 2015, Hamburg, Germany, July 20-21, 2015, Proceedings*, pages 187–198.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, pages 177–190, Berlin, Heidelberg. Springer-Verlag.

Tirthankar Dasgupta and Lipika Dey. 2016. Automatic Scoring for Innovativeness of Textual Ideas. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*.

Lee R. Dice. 1945. Measures of the Amount of Ecologic Association between Species. *Ecology*, 26(3):297302.

Evgeniy Gabrilovich, Susan Dumais, and Eric Horvitz. 2004. Newsjunkie: Providing Personalized Newsfeeds via Analysis of Information Novelty. In *Proceedings of the 13th international conference on World Wide Web*, pages 482–490. ACM.

Tengfei Ma Xiaojun Wan Houping Jia, Xiaojiang Huang and Jianguo Xiao. 2010. PKUTM Participation at TAC 2010 RTE and Summarization Track. National Institute of Standards and Technology Gaithersburg, Maryland, USA.

Paul Jaccard. 1901. Étude Comparative de la Distribution Florale dans une Portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579.

Arun Jayapal. 2012. Similarity Overlap Metric and Greedy String Tiling for Plagiarism Detection at PAN 2012. In Pamela Forner, Jussi Karlgren, and Christa Womser-Hacker, editors, *CLEF (Online Working Notes/Labs/Workshop)*, volume 1178 of *CEUR Workshop Proceedings*, Rome, Italy. CEUR-WS.org.

Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. 2007. Harmonic Coordinates for Character Articulation. *ACM Trans. Graph.*, 26(3), july.

Margarita Karkali, François Rousseau, Alexandros Ntoulas, and Michalis Vazirgiannis. 2013. Efficient Online Novelty Detection in News Streams. In *International Conference on Web Information Systems Engineering*, pages 57–71. Springer.

J Richard Landis and Gary G Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *biometrics*, pages 159–174.

Xiaoyan Li and W Bruce Croft. 2005. Novelty Detection Based on Sentence Level Patterns. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 744–751. ACM.

Bai L. Nguyen H.V. 2011. Cosine Similarity Metric Learning for Face Verification. In *Kimmel R., Klette R., Sugimoto A. (eds) Computer Vision ACCV 2010. ACCV 2010.*, volume 6493 of *Lecture Notes in Computer Science*, pages 709–720, Berlin, Heidelberg. Springer.

Cramer N. Rose S., Engel D. and Cowley W. 2010. Automatic keyword extraction from individual documents. *In M. W. Berry and J. Kogan (Eds.), Text Mining: Theory and Applications: John Wiley and Sons.*

Liyun Ru, Le Zhao, Min Zhang, and Shaoping Ma. 2004. Improved Feature Selection and Redundance Computing-THUIR at TREC 2004 Novelty Track. In *TREC*, Gaithersburg,MD.

Tanik Saikh, Sudip Kumar Naskar, Chandan Giri, and Sivaji Bandyopadhyay. 2015. Textual Entailment Using Different Similarity Metrics. In *Computational Linguistics and Intelligent Text Processing - 16th International Conference, CICLing 2015, Cairo, Egypt, April 14-20, Proceedings, Part I*, pages 491–501.

Tanik Saikh, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. 2016. JU_NLP@DPIL-FIRE 2016: Paraphrase Detection in Indian Languages - A machine Learning Approach. In *Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation, Kolkata, India.*, pages 275–278.

Barry Schiffman and Kathleen McKeown. 2005. Context and Learning in Novelty Detection. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 716–723, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Tomoe Tomiyama, Kosuke Karoji, Takeshi Kondo, Yuichi Kakuta, Tomohiro Takagi, Akiko Aizawa, and Teruhito Kanazawa. 2004. Meiji University Web, Novelty and Genomic Track Experiments. In *TREC*.

M. Tsuchida and K. Ishikawa. 2011. IKOMA at TAC2011: A Method for Recognizing Textual Entailment using Lexical-level and Sentence Structure-level Features. National Institute of Standards and Technology Gaithersburg, Maryland, USA.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Ellen M Voorhees. 2002. Overview of TREC 2002. In *Trec*, National Institute of Standards and Technology Gaithersburg, MD.

Ellen M Voorhees. 2003. Overview of TREC 2003. In *TREC*, pages 1–13, National Institute of Standards and Technology Gaithersburg, MD.

Charles L Wayne. 1997. Topic Detection and Tracking (tdt). In *Workshop held at the University of Maryland*, volume 27, page 28. Citeseer.

Yi Zhang and Flora S Tsai. 2009. Combining Named Entities and Tags for Novel Sentence Detection. In *Proceedings of the WSDM'09 Workshop on Exploiting Semantic Annotations in Information Retrieval*, pages 30–34. ACM.

Yi Zhang, Jamie Callan, and Thomas Minka. 2002. Novelty and Redundancy Detection in Adaptive Filtering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 81–88. ACM.

Min Zhang, Chuan Lin, Yiqun Liu, Leo Zhao, and Shaoping Ma. 2003a. THUIR at TREC 2003: Novelty, Robust and Web. In *TREC*, pages 556–567, Gaithersburg,MD.

Min Zhang, Ruihua Song, Chuan Lin, Shaoping Ma, Zhe Jiang, Yijiang Jin, Yiqun Liu, Le Zhao, and S Ma. 2003b. Expansion-based Technologies in Finding Relevant and New Information: Thu TREC 2002: Novelty Track Experiments. *NIST SPECIAL PUBLICATION SP*, (251):586–590.

Is your Statement Purposeless? Predicting Computer Science Graduation Admission Acceptance based on Statement Of Purpose

Diptesh Kanojia[†,♣,⋆], **Nikhil Wani**[‡,†], **Pushpak Bhattacharyya**[†]
[†]Center for Indian Language Technology, IIT Bombay, India
[♣]IITB-Monash Research Academy, India
[⋆]Monash University, Australia
[†]{diptesh, pb}@cse.iitb.ac.in
[‡]nick.nikhilwani@gmail.com

Abstract

We present a quantitative, data-driven machine learning approach to mitigate the problem of unpredictability of Computer Science Graduate School Admissions. In this paper, we discuss the possibility of a system which may help prospective applicants evaluate their Statement of Purpose (SOP) based on our system output. We, then, identify feature sets which can be used to train a predictive model. We train a model over fifty manually verified SOPs for which it uses an SVM classifier and achieves the highest accuracy of 92% with 10-fold cross validation. We also perform experiments to establish that Word Embedding based features and Document Similarity based features outperform other identified feature combinations. We plan to deploy our application as a web service and release it as a FOSS service.

1 Introduction

Computer Science (CS) graduate admissions process often involves holistic evaluation of prospective applicant based on multiple subjective and quantitative parameters (Ward, 2006). Amongst these parameters the applicant's Statement of Purpose (SOP) serves as a document to convince its readers' *i.e.* the faculty on the selection committee - that one has recorded solid achievements which reflect promise for success in graduate study and hence submission of such a good quality SOP becomes of paramount importance.

Furthermore, Graduate admissions to most Elite universities in the United States of America (USA) only open twice every year - Fall and Spring semesters.

Terminology: We use the terms essay and SOP interchangeably further during our discussion of the work.

2 Motivation

Applicants spend a great deal of time writing SOPs for the admissions process. A well written SOP is a must for an applicant to ensure their admission in any university, and more so for elite universities. Their thoughts and ideas should be organized in their statement. University guidelines[1,2], Alumni blogs[3], and Admission consultancy blogs[4] recommend spending ample time on each SOP and tailoring it to perfection. They also recommend stylometry for writing an essay i.e. word limit, active voice, coherence, and continuity. Various NLP applications like Essay grading (Larkey, 1998), Text Summarization (Gupta and Lehal, 2010) and Sentiment Analysis (Joshi et al., 2015) utilize these features. Hence, we believe that an application that evaluates their statement is crucial. The key question that this paper attempts to answer is:

' *Can information gained from an SOP be used to predict the outcome of a candidate application for graduate school admissions?* '

3 Related Work

Ward (2006) discuss a qualitative model for Graduate Admissions to Computer Science programs but do not use any Machine Learning or Deep Learning based techniques for estimating a likelihood. According to them, other factors which affect the decision of the committee reviewing the applications include Graduate Record Examinations (GRE) score, Undergraduate Grade

[1] http://grad.berkeley.edu/admissions/apply/statement-purpose/
[2] http://admission.stanford.edu/apply/freshman/essays.html
[3] http://alumnus.caltech.edu/~natalia/studyinus/guide/statement/q&a.htm
[4] http://www.happyschools.com/strengthen-your-graduate-school-application/

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 141–145

Point Average (GPA), Letters of Recommendation (LORs), Financial preparation of a candidate, Alignment with institute needs keeping in mind the diversity goals of the university, and lastly the Undergraduate Major of the candidate. They require the user to rate the application parameters and provide ratings as an input to their system. As an output, they provide an estimate of acceptance based on their model[5].

On the other hand, we employ the existing state-of-the-art techniques, identify features and use some of them to predict the acceptance of a candidate. We acknowledge that we do not model all parameters described above.

Another similar study (Raghunathan, 2010) tries to subjectively discuss the admissions process and details the factors which participate in the decision making process of an admission committee. They break the components of a graduate school admissions process and state that SOP is one of the trickiest components of an overall application. They also note that too long an SOP would deter the chances of selection of the candidate. In light of these studies, we focus on creating a model which is able to grade an SOP based on ML techniques.

Text Similarity and related measures (Choi et al., 2010; Adomavicius and Tuzhilin, 2005; Gomaa and Fahmy, 2013) have been extensively studied and used for various NLP applications *viz.* Information Retrieval (Salton et al., 1983), Sense Disambiguation (Resnik and others, 1999). To the best of our knowledge, there is no reported study which evaluates SOPs based on the features identified by us, or use ML and DL based techniques of this kind, at the time of submission. Most of the articles list various parameters which are considered by an admissions committee and a Statement of Purpose (SOP) is a common factor among all.

4 Experiment Design and Setup

In this section, we provide details about our experiment setup and features used for the classification task.

4.1 Dataset

We create our dataset by collecting essays from i) Acquaintances ii) Publicly disclosed SOPs from personal websites, and iii) Admission consultancy blogs. For calculating the similarity measures, we concatenate the essays of the successful applicants, and create a corpus which is used for comparison with both training and testing data.

We collect a total of 50 manually verified SOPs from Elite Universities (low acceptance rate <= 15%) and rejected essays equally split into two sets. We plan to release the dataset publicly under the CC-BY-SA-4.0[6] license.

4.2 Methodology

We use conventional Machine Learning (ML) algorithms (Hall et al., 2009) like Support Vector Machines (SVM) (Vapnik, 2013), Logistic Regression (LR) (Walker and Duncan, 1967), and Random Forest Decision Trees (RFDT) (Ho, 1998) for the task and provide a comparison in Section 5.

We use deep learning approaches and deploy a simple Feed Forward Neural Network to classify the SOPs. We split our data in two folds where the first half is used for training, and the second half is then split into tuning and testing datasets. We also use Multilayer Perceptron, another simple Feed Forward Neural Network (NN) and perform a standard 10-fold cross validation on our dataset. We do acknowledge the modest size of our dataset, but we provide rigorous experimentation including an ablation test to verify that our performance on all classes of our data are unbiased.

4.3 Experiment Design

We cluster the set of features in the following groups - **a) Textual Features** - Feature values based on text contained within the document, **b) Word Embedding based Features** - Features based on average of vector values provided by pre-trained model on Google News Corpora, **c) Similarity Score based and Error based features** - Features based on Document Similarity, and other features based on errors in the document. The last set of features have been identified by us, and are our contribution to the work. We, then, use the algorithms mentioned above to calculate precision, recall and F-Score on each feature set.

We also perform an Ablation test to see which feature set combination is performing the best.

[5] http://www.cs.utep.edu/nigel/estimator/

[6] https://creativecommons.org/licenses/by-sa/4.0/

Classifier	P_{acc}	P_{rej}	$\mathbf{P_{avg}}$	R_{acc}	R_{rej}	$\mathbf{R_{avg}}$	F_{acc}	F_{rej}	$\mathbf{F_{avg}}$
RFDT	0.86	0.79	0.83	0.76	0.88	0.82	0.81	0.83	0.82
LR	0.69	0.83	0.76	0.88	0.60	0.74	0.77	0.70	0.74
SVM	0.89	0.96	**0.92**	0.96	0.88	**0.92**	0.92	0.92	**0.92**
Neural Network Based									
Multilayer Perceptron (Train-Test Split)	-	-	0.82	-	-	0.82	-	-	0.82
Feed Forward NN (FFNN) (Train-Tune-Test Split)	-	-	0.36	-	-	0.60	-	-	0.45

Table 1: Performance of our model on 10-fold cross validation

4.4 System Architecture

Our architecture, shown in figure 1, provides the necessary details about the working of our system. The system takes as input the essay of a prospective applicant, calculates feature values for Similarity Score and Error based features along with Word Embedding based features and predicts an **accept** or **reject** based on the classification model being used.

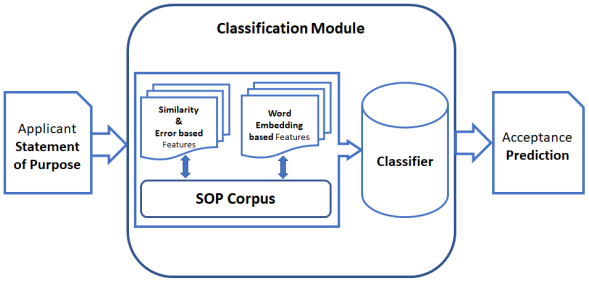


Figure 1: System Architecture

4.5 Features Used

We use the following textual features for evaluating the SOPs. These features have been identified via surveying linguistic properties of a text which may affect the organization and quality of an essay.

4.5.1 Word Embeddings based Features

1. **Average Word Vector Scores** - Average of word vectors of each word in the statement calculated using pre-trained Google News word vectors (Mikolov et al., 2013).

4.5.2 Textual Features

1. **PoS Ratios** - Ratio of nouns, adjectives, adverbs, and verbs to the entire text, obtained using NLTK[7] (Loper and Bird, 2002).

[7] http://www.nltk.org/

	Individual Feature Sets (N-fold)			
Features	2-F	5-F	10-F	50% Split
T [14]	54	46	44	40
WE [300]	48	78	40	44
SE [3]	48	56	56	49
	Combination of Feature Sets			
T + WE [314]	56	62	62	52
T + SE [17]	48	50	38	30
SE + WE [303]	**90**	**92**	**92**	**92**
T + WE + SE [318]	52	50	53	43

Table 2: Ablation test on feature sets using Multi-fold Cross Validation

2. **Discourse Connectors** - It is the number of discourse connectors in the essay computed using a list of discourse connectors[8].

3. **Count of Named Entities** - Number of named entities in the essay. We tried using this as a feature but this drastically lowered the F-scores, and had to be avoided in the final set of reported experiments.

4. **Readability** - The Flesch Reading Ease Score (FRES) of the text (Flesch, 1948).

5. **Length features** - Number of words in the sentence, number of words in the paragraph, and average word length.

6. **Coreference Distance** - Sum of token distance between co-referring mentions.

7. **Degree of Polysemy** - Average number of WordNet (Fellbaum, 2010) senses per word.

4.5.3 Document Similarity Score and Error based Features

1. **Cosine Similarity** - Cosine Similarity Score of an SOP with the corpus of accepted essays dataset, where we ensure that the SOP being compared is not a part of the accepted essay corpus.

[8] http://www.cfilt.iitb.ac.in/cognitive-nlp/

2. **Similarity-based features using GloVe** - The similarity between every pair of content words in adjacent sentences. The similarity is computed as the cosine similarity between their word vectors from the pre-trained GloVe word embeddings (Pennington et al., 2014). We calculate the mean and maximum similarity values.

3. **Spell Check Errors** - We use PyEnchant[9] to embed a spell checker and count the number of errors in each document. The count is then used as another feature for training classifier.

4. **Out of Vocabulary Words** - We use the pre-trained Google news word embeddings and find out word vectors for every token in the document. The tokens which do not return any vector are either rare words or in all probability out of vocabulary words. We use the count of such tokens as another feature set.

5 Results

We perform the experiments detailed in section 4.3 and report our results on 10-fold cross validation. Among the experiments we perform, we achieve the highest F-score of 92% using the SVM classifier with an RBF Kernel. The results are shown in table 1 and discussed in Section 6.

Table 1 clearly indicates that SVM outperforms Random Forest Decision Trees (RFDT) with a margin of 9%, Logistic Regression (LR) with a margin of 18%, Neural Network based Multilayer Perceptron with a margin of 10%, and another Feed Forward Neural Network (FFNN) with a margin of 47%. We further discuss the impact and justifications of these results in Section 6.

We also perform a multi-fold ablation test, using SVM Classifier, on the feature sets identified in section 4.3. The results for the ablation test are shown in Table 2. The table clearly identifies that Similarity Scores and Error based features along with Word Embedding based features give us the best results.

6 Discussion

In order to identify the features that contribute to the modeled non-linearity of SVM and our best reported accuracy of 92%, we conduct a comprehensive ablation test. Feature sets mentioned in Section 4.3 were considered. A total of 317 features were ablated based on their sets via multi-fold stratified cross validation experiments and additionally in an experiment with 50% split of the dataset as shown in the Table 2.

It was found that the 14 identified Textual (T) features do not contribute significantly to our model. We extrapolate that these features may have worked better in another context such as Sentiment Analysis (Mishra et al., 2017), or Essay Grading (Valenti et al., 2003), but not for the task of SOP Classification. Our task primarily aims at labeling an SOP with an accept or reject, however, we observe that Textual features do not differentiate well between coherent and incoherent essays. We also observe that Word Embedding (WE) features of 300 dimensions contribute significantly towards the accuracy of our final model. While they do not contribute notably when used to perform classification independently, combining them with Similarity Score and Error Based (SE) feature set form our best reported model i.e. SE + WE.

7 Conclusion and Future Work

In this paper we demonstrate the applicability of a data driven approach to mitigate the unpredictability of Computer Science graduate admissions process. We build a corpus of fifty manually verified SOPs from Accepted applicants to Elite Universities (low acceptance rate <= 15%) rejected SOPs. We show that a combination of Cosine Similarity, Error based features and Word Embedding based features outperform any of the textual features based combinations, for this task. Based on the ablation tests conducted, we model an SVM classifier that predicts with significantly high accuracy.

In future, we plan to integrate Parts-of-speech (POS) based similarity measures and Recurrent Neural Networks (RNN) (Cho et al., 2014) which have been shown to work well with textual data. Integration of other traditional metrics of a candidates application performance measure such as GRE, Test of English as a Foreign Language (TOEFL) / International English Language Testing System (IELTS) score and GPA will further robustly extend this model. We also plan to translate this novel research to an open source web application which would allow prospective applicants to evaluate their SOPs with our system.

[9] http://pythonhosted.org/pyenchant/

References

Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Seung-Seok Choi, Sung-Hyuk Cha, and Charles C Tappert. 2010. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48.

Christiane Fellbaum. 2010. Wordnet. *Theory and applications of ontology: computer applications*, pages 231–243.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Wael H Gomaa and Aly A Fahmy. 2013. A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13).

Vishal Gupta and Gurpreet Singh Lehal. 2010. A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence*, 2(3):258–268.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Tin Kam Ho. 1998. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *ACL (2)*, pages 757–762.

Leah S. Larkey. 1998. Automatic essay grading using text categorization techniques. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 90–95, New York, NY, USA. ACM.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*, pages 63–70. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2017. Leveraging cognitive features for sentiment analysis. *arXiv preprint arXiv:1701.05581*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Karthik Raghunathan. 2010. Demystifying the american graduate admissions process. *StudyMode. com*.

Philip Resnik et al. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *J. Artif. Intell. Res.(JAIR)*, 11:95–130.

Gerard Salton, Edward A Fox, and Harry Wu. 1983. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036.

Salvatore Valenti, Francesca Neri, and Alessandro Cucchiarelli. 2003. An overview of current research on automated essay grading. *Journal of Information Technology Education: Research*, 2(1):319–330.

Vladimir Vapnik. 2013. *The nature of statistical learning theory*. Springer science & business media.

Strother H Walker and David B Duncan. 1967. Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2):167–179.

Nigel Ward. 2006. Towards a model of computer science graduate admissions decisions. *JACIII*, 10(3):372–383.

Natural Language Programming with Automatic Code Generation towards Solving Addition-Subtraction Word Problems

Sourav Mandal
Haldia Institute of Technology,
India
sourav.mandal@hithaldia.in

Sudip Kumar Naskar
Jadavpur University,
India
sudip.naskar@cse.jdvu.ac.in

Abstract

Solving mathematical word problems by understanding natural language texts and by representing them in the form of equations to generate the final answers has been gaining importance in recent days. At the same time, automatic code generation from natural language text input (natural language programming) in the field of software engineering and natural language processing (NLP) is drawing the attention of researchers. Representing natural language texts consisting of mathematical or logical information into such programmable event driven scenario to find a conclusion has immense effect in automatic code generation in software engineering, e-learning education, financial report generation, etc. In this paper, we propose a model that extracts relevant information from mathematical word problem (MWP) texts, stores them in predefined templates, models them in object oriented paradigm, and finally map into an object oriented programming (OOP)[1] language (JAVA) automatically to create a complete executable code. The codes are then executed automatically to output the final answer of the MWP. The proposed system can solve addition-subtraction type MWPs and produced an accuracy of 90.48% on a subset of the standard AI2 arithmetic questions[2] dataset.

1 Introduction

Solving MWPs is a very longstanding research problem; researchers in the field of Artificial Intelligence (AI), machine learning and NLP have proposed various methodologies for solving MWPs since 1960s. Word problems are formed using natural language text rather than in mathematical notations (Verschaffel et al., 2000) and they can be of any type of numerical problems based on domains like mathematics, physics, geometry, etc. (Mukherjee and Garain, 2008). Addition-subtraction type MWP is an integral part of basic understanding of mathematics and elementary school level curriculum. The objective of the work presented here is primarily to generate computer programs automatically from natural language texts which when executed will produce the desired answer. Comparison with existing MWP solvers are not appropriate to this kind of work as our end objective is not exactly to create the equation and solve the problem itself, but rather to generate a computer program to solve the problem and thus it adds a new dimension to research in solving MWPs. For example, "*Dan has 64 violet marbles, he gave Mary 14 of the marbles. How many violet marbles does he now have?*" is a word problem which is related to the subtraction or addition operation. This particular problem can be solved manually by noticing the structure of the problem statement in which the first sentence indicates an 'assignment' operation and the second sentence indicates a 'subtraction' operation associated with the verb 'give' for Dan (*primary_owner*) and 'addition' operation for Mary (*secondary_owner*). Final answer requirement is related to violet (*attribute*) marble (*item*) in the possession of Dan. The answer to this problem is simply obtained by subtracting 14 from 64, i.e., 64-14=50.

In the OOP approach, we define 'classes' to represent real life entities and declare instances of those classes called 'objects'. To solve such problems, a computer programmer basically defines a class – 'Person' with the data fields like *name*, *item_name*, *item_attribute* and *item_quantity*, and

[1] http://docs.oracle.com/javase/tutorial/java/concepts/

[2] http://allenai.org/data.html

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 146–154

a method, e.g., *evaluate_result*(). Then he declares objects 'obj1' and 'obj2' of this class. Therefore, for the said example,
$obj1.name = Den$,
$obj1.item\_name = marble$,
$obj1.item\_attribute = violet$ and
$obj1.item\_quantity = 64$
$obj2.name = Mary$,
$obj2.item\_name = marble$,
$obj2.item\_attribute = violet$ and
$obj2.item\_quantity = x$ (not given).
The operation associated with the verb 'has' is '=' (assignment or observation) and can be coded as $obj1.item\_quantity = 64$. The operations associated with the verb 'give' are both '-' and '+' i.e., subtraction and addition (negative_transfer) and can be coded as
$obj1.item\_quantity = obj1.item\_quantity - 14$ and $obj2.item\_quantity = obj2.item\_quantity + 14$.
The arithmetic operators are selected based on the verb categories (cf. Table 2) they belong to and the operations can be executed from within a method, e.g., 'evaluate_result()'. In the present work, the verb categorization is rule-based and is determined from the verb predicates (cf. Table 1).

The system first extracts and stores all the required information for the key entities – owners, items, attributes, quantities, and the arithmetic operations relevant to the verb semantics from the MWP text. Then the system creates composite object entities resembling each unique owner-item-attribute combination in the MWP, finds their states and corresponding state transitions (if any) on the basis of the operations or activities (verbs they face) in that MWP, and generates the relevant computer codes. However, automatic extraction of information from natural language text and computer code generation are not trivial. Moreover, solving MWPs requires natural language understanding and reasoning which are very difficult and most of the research in natural language processing (NLP) tend to do away with it. Therefore, solving MWPs automatically has remained an open research challenge.

However, presently our system is unique in three ways. Firstly, our system tries to capture how a programmer can solve an MWP problem using a JAVA like language and it acts as a bridge between unstructured natural language and structured formal language(s). This transformation from natural to formal language (executable program) throws immense challenges in the field of NLP and Information Extraction (IE). Secondly, OOP paradigm is used to model real world data driven tasks and operations. Word problems are apt to be modeled with OOP since it contains real world entities and their specific activities, which motivated us to use an object oriented approach for the present work. The mathematical equation formation is not important here as all operations are represented with JAVA programming statements which determine the mathematical expressions. Once the desirable complete JAVA program (cf. Figure 3) is formed automatically, rest of the activities like compilation and execution of the program to process the result, are handled by the JAVA compiler itself like any computer language programming assignment and here lies the advantage of the proposed approach. Finally, the proposed approach keeps track of all the entities and their state transitions throughout the text (cf. Figure 2) which makes it much easier to answer any question based on the text, not just the question actually present in the MWP problem. It does not have to start processing afresh for answering any other question based on the same text.

The remainder of the paper is organized as follows. Section 2 presents an overview of relevant related work. Section 3 provides a detailed discussion on the system components. Section 4 outlines the datasets, experiments and the corresponding results together with some analysis, followed by conclusions and avenues for further research in Section 5.

2 Related Work

The research problem on generation of executable computer programs for solving MWPs has not been attempted so far to the best of our knowledge. However, formal language modeling from natural language text has been studied previously in various domain by researchers mainly in software engineering (Bryant et al., 2003; Lei et al., 2013), web interfaces of databases (Alexander et al., 2013), etc. Some researchers tried to represent natural language texts using regular expressions (Kushman and Barzilay, 2013). Ballard and Biermann (Ballard and Biermann, 1979) proposed a natural language computing ('NLC') prototype to process and evaluate small natural language text word problems based on matrix com-

putation. They proposed a method to generate solution from a matrix entry and solve problems like "*add five with the second positive entry in row 5*", "*double the fifth entry and add that to the last entry of that row*", etc. Each of these assignments have some types of mathematical terminologies like 'add', 'double', etc., which clearly indicates the operation or operator. This research problem is not exactly related to automatic program generation, rather it is about processing a matrix data structure syntactically to generate the desired result based on matrix arithmetic. Liu and Lieberman (Liu and Lieberman, 2005) developed a system 'Metafor' which converts a small description of an event into a 'Python' program based on interaction logs with respect to time and entity participation. Kate et al. (Kate et al., 2005) tried to represent natural language texts syntactically and semantically into a formal representation that is based mainly on deterministic context-free grammar. They used "if-then" rules to develop a new formal language 'CLANG' for processing natural language text. Mihalcea et al. (Mihalcea et al., 2006) first proposed a system that attempts to convert natural language texts directly into computer programs. They tried to identify various algorithmic steps, decisions and loop structures from English text representing any event and convert it into a program skeleton using 'PERL' programming language which is object oriented in nature. Following the "who_does_what" structure their system develops a program skeleton and generates the 'PERL' code for texts like "*When customer orders a drink, the bartender makes it*". They developed a model which creates different classes like 'Customer', 'Bartender' and relevant methods like 'order_drink()', 'make_drink()' to support their actions. Our work is little relevant to their work. Alongside, many researchers proposed various methodologies to solve MWPs (Kushman et al., 2014; Hosseini et al., 2014; Walker and Kintsch, 1985; Fletcher, 1985; Roy and Roth, 2015; Shi et al., 2015; Mitra and Baral, 2016). The work presented in this paper differs from these works.

3 System Description

3.1 Mapping Input Texts to The Concept

Natural language texts representing some MWPs typically contain multiple factual sentences and a 'question sentence' at the end (cf. the example given in Section 1). Each sentence may or may not have some mathematical meaning. Our objective is to identify the key players or entities and their state transitions from the first sentence they occur in and till the last sentence. An MWP example containing multiple sentences is given below.

> *"Harry has 15 blue and 10 green balloons. He lost 5 blue balloons in the market. Then he bought 3 green balloons from a shop. Tim has 12 kites and 10 blue balloons. Tim gave Harry 4 blue balloons.. ... How many green balloons does Harry have? "*

This MWP problem involves 2 persons having 2 types of balloons, blue and green, and 1 person having kites. Here owner entity names are 'Harry', 'Tim', and item entity names are 'balloon' and 'kite', and item attributes associated with the item 'balloon' are 'blue' and 'green'. Our objective is to map such information expressed in natural language texts into object oriented programming paradigm. Every sentence is considered as a state and throughout the input text several state transitions take place with all unique 'Owner–Item–Attribute'(OIA) objects (cf. Figure 2). Here we create objects like 'Harry-balloon-blue', 'Harry-balloon-green', 'Tim-balloon-blue', 'Tim-kite-null' along with their respective quantities. It is to be noticed that the owner does not have to be a person always. Our system identifies all different types of owner, item (and attribute, if any) combinations from the input text and create 'objects' for each of them. It also identifies their state transitions that they go through throughout the problem text. Most importantly, if the text has question sentence like *"How many blue balloons are now with Harry?"* or *" How many kites does Tim have now?"*, the system formulates the answer by matching the 'OIA' object in the question sentence. Our system carries the information about all the 'OIA' objects and the changes in quantities of the items (if any) occurring in association with the operations (related to the 'verbs') they face till their final state. Therefore, after processing the question sentence and identifying the 'OIA' object associated with it, the system displays the final processed quantity of the corresponding 'OIA' object as the answer.

3.2 MWP Text Simplification

To make the processing more convenient, the input text is simplified first. Conjunctions are re-

moved and coreferences are substituted to convert the input text into a simplified format so that we can extract information with out any ambiguities. We use Stanford CoreNLP[3] suite 3.6.0 to perform the intermediate NLP tasks, e.g., POS tagging, dependency parsing, coreference resolution, etc., and extract relevant information. We remove conjunctions like 'and', ',' (comma), 'but', ', and' and ', but' from compound sentences and break them into multiple simple sentences. The coreference mentions for pronouns like 'he', 'she', 'his', 'her' etc., are substituted with the corresponding referred expressions so that we can extract the owner entities directly and unambiguously.

3.3 Information Extraction based on Semantic Role Labelling (SRL)

SRL techniques are mainly used to semantically process texts and to define role(s) of every words present in a text. For extracting information from text, we used the SRL tool – Mateplus[4] (Roth and Woodsend, 2014; Roth and Lapata, 2015), which was developed for meaning representations based on the CMU SEMAFOR[5] tool and frameNet[6]. Table 1 shows the output of 'Mateplus' for the sample sentence "*Sam gave Mary 23 green marbles.*". Depending on the type of the predicates and also

ID	Form	POS	Dependency	Predicate	Args:Locating
1	Sam	NNP	SUB	-	Donor
2	gave	VBD	ROOT	Giving	-
3	Mary	NNP	OBJ	-	Recipient
4	23	CD	NMOD	-	-
5	green	JJ	AMOD	-	-
6	marbles	NNS	OBJ	-	Theme
7	'.'	'.'	P	-	-

Table 1: A sample SRL output

the verb grouping from VerbNet[7], the verbs are manually categorized and respective equations are generated by the system (cf. Subsection 3.4). Given the example, the predicate (e.g., 'Giving'), owners (e.g., 'Donor' as primary and 'Recipient' as secondary owner), items (e.g., 'Theme') and the attribute(s) of the item(s) are extracted from the SRL output (cf. 'give', 'Sam', 'Mary','marble' and 'green' in Table 1 respectively). Depending on the type (i.e., category) of the predicates, respective 'operations' are identified for each 'OIA' triplet/object (cf. Section 3.4).

The system extracts all relevant information from the input MWP texts, sentence by sentence, identifying the owners, items, item attributes, 'verb', 'cardinal number' (or 'quantity') from the SRL output (cf. Table 1) using a rule-based approach. These information are extracted from the POS tag, and dependency relations combined with 'predicates' and relevant 'arguments'. For example, *NNP/NN* and *SUB* is an 'owner' entity, *NNP/NN* and *NMOD/PMOD/OBJ* is a 'secondary owner', *NNS/NN* and *OBJ* is an 'item', *JJ* and *AMOD/NMOD* is an 'item-attribute'. A maximum of 5 conditions (rules) are used to identify each type.

3.4 Verb Categorization & Equation Formation

We studied the verbs appearing in the dataset (cf. Section 4) and by manually analyzing the predicates and arguments (cf. Subsection 3.3 and Table 1), we grouped the verbs into 5 categories based on the frameNet frame definitions along with the similarity of the verbs in terms of VerbNet verb grouping and probable arithmetic operational connotation (=, +, -) as in Table 2. We carried out verb categorization motivated by the work of (Hosseini et al., 2014).

Category	Verbs	Operator
Observation	$have, find$	$assignment$
Increment	$gather, grow$	$+$
Decrement	$lose, spend$	$-$
Positive Transfer	$take, receive$	$+ and -$
Negative Transfer	$give, sell, pay$	$- and +$

Table 2: Schema and operations for the verb categories.

For example, using the frame definition of 'Giving[8]' in the 'frameNet', we categorized 'give' in the 'negative transfer' category where '-' operator is associated with the donor/primary_owner (*Sam* in Table 1) and '+' operator is associated with the recipient/secondary_owner (*Mary* in Table 1). The frame definition for 'Giving' is (*Donor, [Recipient], Theme/Items, [Quantities], [Time], [Location]....*). Similarly, we categorized

[3] http://stanfordnlp.github.io/CoreNLP/

[4] https://github.com/microth/mateplus

[5] www.ark.cs.cmu.edu/SEMAFOR

[6] https://framenet.icsi.berkeley.edu

[7] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html

[8] https://framenet2.icsi.berkeley.edu/fnReports/data/frameIndex.xml?frame=Giving

Category	Examples	Schema Entry	Equations
(Null) Observation	Joan has 40 blue balloons	[Joan, null, balloon blue, 40]	Joan-balloon-blue.quantity=40
Increment	Tom grew 9 watermelons	[Tom, null, watermelon, null, 9]	Tom-watermelon-null.quantity=Tom-watermelon-null.quantity+9
Decrement	Sally lost 2 of the orange balloons	[Sally, null, balloons, orange, 2]	Sally-balloon-orange.quantity= Sally-balloon-orange.quantity-2
Positive Transfer	Dan took 22 pencils from the drawer	[Dan, drawer, pencils, null, 22]	Dan-pencil-null.quantity=Dan-pencil-null.quantity+22 and drawer-pencil-null.quantity=drawer-pencil-null.quantity-22
Negative Transfer	Jason gave 13 of the seashells to Tim	[Jason, Tim, seashell, null, 13]	Jason-seashell-null.quantity=Jason-seashell-null.quantity-13 and Tim-seashell-null.quantity=Tim-seashell-null.quantity+13

Table 3: Equation formation based on verb category and schema information

the verbs like 'has', 'find', 'are' having similar kind of frame definitions in the 'observation' category representing the '=' operation as they do not refer any changes. We developed a database schema to store the extracted information from each input sentence by analyzing the predicates associated with the verbs contained in the MWPs. The schema is generic and defined as *[primary_owner, secondary_owner, item_name, item_attribute, item_count]* for all categories of verbs that could be present in the input text sentences. Table 2 presents the verb categories (based on only one sense of the verbs) and the corresponding related operations. The 'Positive Transfer' and 'Negative Transfer' categories represent two operators connected with the 'primary_owner' and 'secondary_owner'.

Table 3 presents the targeted equations related to the verb categories. The schema entry (cf. Table 3) includes extracted information for primary_owner, secondary_owner, item_name, item_attribute, item_quantity from each sentence in the input MWP text. Owner_name (primary_owner or secondary_owner), item_name and item_attribute, these 3 components create a single 'OIA' entity throughout the input text processing. E.g., the sentence "*Jason gave 13 of the seashells to Tim"* contains the primary_owner 'Jason', secondary_owner 'Tim', item_name 'seashell' and item_attribute 'null (no attribute)' (cf. Table 3). Here, 'Jason-seashell-null' and 'Tim-seashell-null' can be referred as two 'objects', say 'Object[0]' and 'Object[1]', in the OOP scenario where the 'item_quantity(i.e. 13) is subtracted (i.e., -) from 'Object[0]' and added to 'Object[1]' since the verb 'give' belongs to the 'Negative Transfer' category. Similarly, the sentence "*Joan has 40 blue balloons*" will create an object entity 'Joan-balloon-blue' where the 'OIA' object is associated with the 'assignment' ('=') operator with the 'item_quantity'(i.e. 40). The equation formations also follow the same directions as in Table 3. An input text sentence may not always have 'secondary_owner' or 'item_attribute'. Therefore, some of the schema entries are shown as 'null' in Table 3.

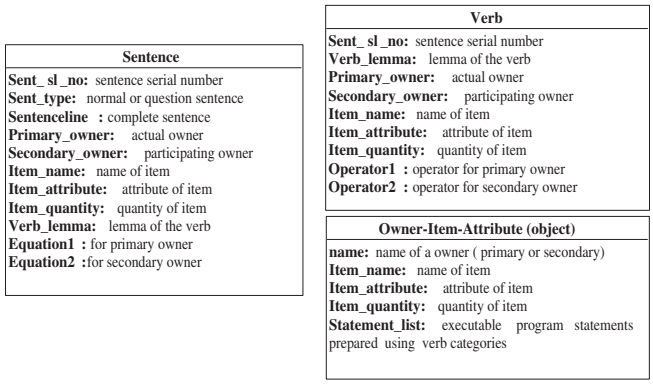


Figure 1: Template-based information extraction.

3.5 Information Processing & Template Filling

We followed a template based IE approach and used three templates – 'Sentence', 'OIA' and 'Verb'. Figure 1 describes the three templates. After extracting all the relevant information, the system stores them in the 'Sentence' and 'Verb' templates. Successively, the system identifies each unique 'OIA' triplet such that at least one component is unique with respect to the other triplets (cf. Section 3.6). This procedure creates a number of instances of 'OIA' template based on the identified unique triplets. Then by processing the extracted information in the 'Verb' template, the desired equation(s) is generated with the associated 'OIA' triplet(s) according to the verb category (cf. 'Equation' column in Table 3) for each sentence. The generated equations are then added in the 'sentence' template as 'Equation1' (for primary owner) and as 'Equation2' (for secondary_owner) (cf. Figure 1). Finally, real programming 'objects' are created related to all the 'OIA' triplets. By matching and replacing the

Owner-Item-Attribute / Objects	Item_ count(x)	Verb_ lemma	Operation	Equation statements	State no. / Sentence no.
mike-balloon-orange / obj[0]	8	have	assignment	obj[0].quantity=8	1 / 1
Sam-balloon-orange / obj[1]	14	have	assignment	obj[1].quantity=14	1 / 2
mike-balloon-orange / obj[0]	4	give	-	obj[0].quantity= obj[0].quantity-4	2 / 3
Sam-balloon-orange / obj[1]	4	give	+	obj[1].quantity= obj[1].quantity+4	2 / 3

Table 4: Generating program statements

triplets with the respective actual objects in the 'equations' (cf. 'Equations' column in Table 3) of the 'Sentence' templates, the actual JAVA programming statements are created (cf. Subsection 3.6 and Table 4). These program statements are then appended according to the sequence of occurrence of the 'OIA' objects in a MWP following their state diagrams (cf. Figure 2), which make up the executable JAVA program (cf. Figure 3). The system generates 'Sentence' templates equal to the number of sentences in an MWP, 'Verb' templates equaling the numbers of verbs existing, and an 'OIA' template for each unique 'OIA' object. The last 'Sentence' template is the 'question sentence' which is separately analyzed to identify question requirements. Extracted and processed information stored in templates are finally stored in tables using a relational database approach – MYSQL[9]. This introduces structure into the unstructured natural language texts and also makes the processing and reasoning tasks easier.

3.6 Automatic Program Generation Using OOP Approach

3.6.1 Object Creation

In order to generate an object oriented program from the input text, the first task is to represent the identified unique 'OIA' combinations as real 'Objects' in OOP. E.g., if after simplification, the input MWP text is "*Mike has 8 orange marbles. Sam has 14 orange marbles. Mike gave Sam 4 of the marbles. How many orange marbles does Mike now have?*", the system identifies 2 unique 'OIA' combinations – 'Mike-marble-orange' and 'Sam-marble-orange'. The identified 'OIA' triplets are then represented as 'obj[0]' and 'obj[1]' using the OOP concept. These objects are the real instantiation of the predefined class 'OwnerItem' (cf. Figure 3) resembling an 'OIA' template. The system dose not consider the question sentence for object creation.

3.6.2 JAVA Program Statements Generation

For the example in Section 3.6.1, the verb, 'have', belongs to the observation category (cf. Table 2) and therefore generate the assignment ('=") statements. The verb 'give' is the type of 'negative transfer' (cf. Table 2)and it produces the statement having subtraction operation ('-') with the primary_owner 'Mike' (obj[0]) and addition operation ('+') with the secondary_owner 'Sam' (obj[1]), shown in the 'Equation statements' column in Table 4. The statements as a whole lead to the executable program statements in JAVA language. Table 4 shows, how the 'OIA' objects are created, corresponding values are associated to the objects, the mathematical operations are identified from the verb lemma and the corresponding program statements are generated from the same example. The equations are first formed, e.g., "*mike-balloon-orange.quantity=8*" (like 'Equations' column in Table 3) in which the 'OIA' objects are later replaced by real objects and new equations are formed, e.g., "*obj[0].quantity=8*" (cf. 'Equation statements' column in Table 4) using JAVA programming syntaxes.

3.6.3 State Transition Diagram

Figure 2 demonstrates a simple forward "state transition diagram" for all 'OIA' combinations or resultant 'OwnerItem' objects for the example mentioned earlier. A 'state' of an 'OwnerItem' objects is basically the sentence in the MWP texts where it exists. An 'OwnerItem' object in the program appears first in any one of the sentences

[9] http://www.mysql.co\ref{tab:VerbCategories}.m/

(first state) and moves towards the last sentence they appear in (last state)of a MWP except the question sentence (referred as forward transition). The question sentence does not result in any state change. Generally it does not have any operation associated with it. Therefore, individual 'Owner-

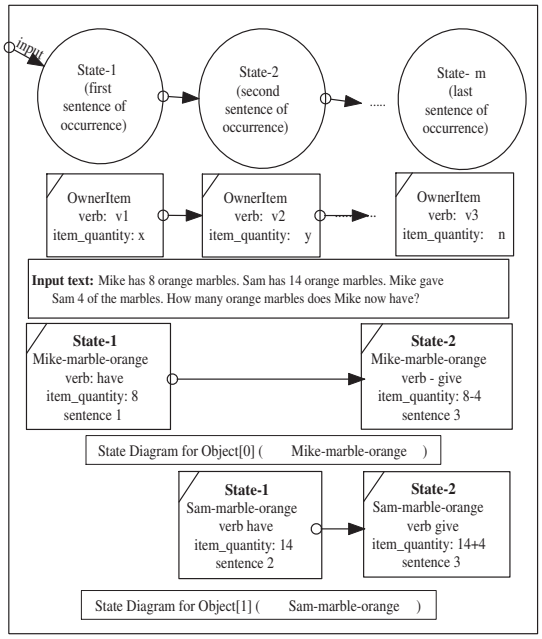


Figure 2: State diagrams of the 'OwnerItem' objects

Item' object entities have their own state transition diagrams based on their presence in the sentences. In every sentential state they may participate in an operation or not. In Table 4, the 'objects' obj[0] has 2 states, occurring in sentence numbers 1 and 3 and obj[1] has 2 states occurring in sentence numbers are 2 and 3. Figure 2 gives a pictorial representation of the sentential states of the object entities and their forward transitions based on the operations they performed. The last state of any objects are having the final quantity associated to them. Analyzing the extracted information from question sentence the object is identified for whom the answer will be displayed (obj[0] in the Figure 2). The state diagram in Figure 2 is related to the CHANGE type word problems (Mitra and Baral, 2016) having all quantities available for the desired object in terms of answer generation. In some cases, where the problems have an adverbial modifier like all, total, together, etc (COMBINE type (Mitra and Baral, 2016)), it is observed that each unique object has single state and no transition. In such scenario, the statements (or related state quantities) of all relevant objects are summed up to generate complete JAVA code which produces final answer.

3.6.4 Executable Program Generation

After creation of the program statements for all individual 'OIA' objects, they are integrated in a predefined JAVA program skeleton in a rule-based manner. The desired program statements are only considered and added according to the sequence of occurrence (i.e., events) in the given MWP. Figure 3 shows the generated program for the same example text. The system processes the question sentence to extract information about the 'OwnerItem' about whom (or which) the question has been asked and the presence of any modifier like 'all', 'total' etc (indicates summation). Subsequently, the extracted information is used to generate additional program statements (to be appended at the end and not given in Figure 3) that processes and displays the desired final answer. After the program generation, compilation and execution of that program are performed by the JAVA compiler (JVM) itself to generate the final answer.

Input text - Mike has 8 orange marbles. Sam has 14 orange marbles. Mike gave Sam 4 of the marbles.

Generated Program-

```
class OwnerItem { //class template
public String owner_name;
public String item_name;
public String item_attribute;
public int item_count = 0;
public void setname (String name, String var,
String atr ) {
owner_name = name;
item_name = var;
item_attribute = atr ;
}
public void display() {// display states
System.out.println ("Owner is:" + owner_name);
System.out.println ("Item is:" + item_name);
System.out.println ("Item attribute is:" +
item_attribute);
System.out.println ("Count:" + item_count);}}
```

```
public class Evaluation { //main program
for execution
 public static void main(String args []) {
int total_owner=2;
int x=0;
// array of objects
OwnerItem obj [] = new
OwnerItem [total_owner];
for ( int i = 0; i < obj.length ; i++) {
obj [i] = new OwnerItem (); //object creation
}
obj [0].item_count=8;
obj [0].item_count= obj [0].item_count-4;
obj [1].item_count=14;
obj [1].item_count= obj [1].item_count+4;
obj [0]. setname ("mike","balloon","orange");
obj [0].display();
obj [1]. setname ("Sam","balloon","orange");
obj [1].display();}}
```

Output- mike-balloon-orange (obj [0])= 4 (answer); sam -balloon-orange(obj [1])= 18

Figure 3: Automatically Generated Program

4 Dataset, Results and Discussions

There is a broad sense of natural language programming available in literature, however, they do not exactly relate to our objective or methodology. Though no standard datasets are available specifically for such work, we compiled a dataset containing 189 questions. We selected word problems from the dateset available with the work of (Hosseini et al., 2014) which is the same as the 'AI2 Arithmetic Questions' dataset. They compiled 395 addition-subtraction word problems with 3 subsets

– MA1, IXL, and MA2 with varying degree of complexity. Our selection was based on the constraint that the sentences of each word problems must have links between them towards the forward movement of state transitions and each sentence in a MWP (i) must not have any "missing information" and (ii) must not be an "irrelevant sentence" with respect to answer generation. For example, the problems "*Joan found 70 seashells on the beach. She gave Sam some of her seashells. She has 27 seashells. How many seashells did she give to Sam?*" contains a sentence having the word 'some' which does not hold any definitive cardinal value, instead indicates a operation, are referred to as 'missing information'. Another example from the dataset is "*Tom purchased a Batman game for $ 13.60 , and a Superman game for $ 5.06. Tom already owns 2 games. How much did Tom spend on video games?*. In this example, the sentence "*Tom already owns 2 games.*" does not have any actual relation with the desired result and this kind of sentences are referred to as 'irrelevant sentence'. These cases were not included in the dataset since our system presently does not have the capabilities to handle them. We selected in total 189 problems[10] from MA1 and MA2 (out of total 255 problems) based on the constraints. We did not consider IXL since the corresponding problems involve more information gaps which call for complex reasoning (due to ambiguities in owners, items) that can not be handled by the proposed approach and some problems of MA1 or MA2 do not fit with our objective.

The system generated syntactically correct programs in all cases, however, in terms of correct answer generation (i.e., logically correct programs) it produced an accuracy of 90.48% (171 out of 189) on the test dataset. The system performed properly for texts containing CHANGE or COMBINE information. Cases for which the system did not produce correct results are given below with some examples.

- **No Link Among Owners, Participating Operation:** E.g., Dan had 7 potatoes and 4 cantaloupes in the garden. The rabbits ate 4 of the potatoes. How many potatoes does Dan now have? (8 problems/44.5%)

- **Wrong Program/ Answer Generation Due to Various Reasons like Program Logical Errors, Sentence Sequence, Wrong IE/ SRL etc.:** E.g., There are 7 crayons in the drawer and 6 crayons on the desk . Sam placed 4 crayons and 8 scissors on the desk . How many crayons are now there in total ? (7 problems/38.9%)

- **Improper Reasoning of Question Sentence:** E.g., A restaurant served 9 hot dogs during lunch and 2 during dinner today. It served 5 of them yesterday. How many hot dogs were served today? (3 problems/16.6%)

[10]dataset available at: https://sites.google.com/site/autocodegeneration/

5 Conclusions

Object oriented analysis and design approach is very useful for modeling any real world data and event-driven scenario with ease. We only need to identify the key entities and their roles in that scenario. The main objective of our work is the generation of structured programs (JAVA based) automatically from natural language MWP texts, not exactly the solution of the MWPs itself, which can be further extended to become a complete MWP solver. The work is more relevant to natural language programming (like Mihalcea et al. (2006)) rather than the development of an MWP solver. We selected the MWP domain since it is event-driven and involves operations like assignment, addition, subtraction, etc., related to the associated verbs. We tested our system on typically small input texts containing only 3–4 sentences (i.e., before text simplification), however, the approach is generic and it will also work for longer input texts. The approach can also be extended for potential use in question answering and summarization purposes by identifying the key players like owners and items for the domains that handle operations like additions and subtractions. If we augment the model with various 'OIA' entities and large number of functionalities then the methodology can represent any natural language text specific to some domain into an object-oriented paradigm and can add great power to automatic code generation from software requirement specifications and software designs. We would also like to extend the proposed OOP based approach to model and solve word problems involving multiplication and division and try to minimize the constraints.

Acknowledgments

Sudip Kumar Naskar is supported by Media Lab Asia, MeitY, Government of India, under the Young Faculty Research Fellowship of the Visvesvaraya PhD Scheme for Electronics & IT.

References

Rukshan Alexander, Prashanthi Rukshan, and Sinnathamby Mahesan. 2013. Natural language web interface for database (NLWIDB). *CoRR*, abs/1308.3830.

Bruce W. Ballard and Alan W. Biermann. 1979. Programming in natural language: "NLC"; as a prototype. In *Proceedings of the 1979 Annual Conference*, ACM '79, pages 228–237, New York, NY, USA. ACM.

Barrett R Bryant, Beurn-Seuk Lee, Fei Cao, Wei Zhao, and Jeffrey G Gray. 2003. From natural language requirements to executable models of software components. Technical report, DTIC Document.

Charles R Fletcher. 1985. Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers*, 17(5):565–571.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 523–533.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 1062–1068.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 826–836.

Nate Kushman, Luke Zettlemoyer, Regina Barzilay, and Yoav Artzi. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 271–281.

Tao Lei, Fan Long, Regina Barzilay, and Martin C. Rinard. 2013. From natural language specifications to program input parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1294–1303.

Hugo Liu and Henry Lieberman. 2005. Metafor: visualizing stories as code. In *Proceedings of the 2005 International Conference on Intelligent User Interfaces, January 10-13, 2005, San Diego, California, USA*, pages 305–307.

Rada Mihalcea, Hugo Liu, and Henry Lieberman. 2006. NLP (natural language processing) for NLP (natural language programming). In *Computational Linguistics and Intelligent Text Processing, 7th International Conference, CICLing 2006, Mexico City, Mexico, February 19-25, 2006, Proceedings*, pages 319–330.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artif. Intell. Rev.*, 29(2):93–122.

Michael Roth and Mirella Lapata. 2015. Context-aware frame-semantic role labeling. *TACL*, 3:449–460.

Michael Roth and Kristian Woodsend. 2014. Composition of word representations improves semantic role labelling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 407–413.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1743–1752.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1132–1142.

Lieven Verschaffel, Brian Greer, and Erik De Corte. 2000. *Making sense of word problems*. Lisse Swets and Zeitlinger.

William H. Walker and Walter Kintsch. 1985. Automatic and strategic aspects of knowledge retrieval. *Cognitive Science*, 9(2):261–283.

Unsupervised Separation of Transliterable and Native Words for Malayalam

Deepak P
Queen's University Belfast, UK
deepaksp@acm.org

Abstract

Differentiating intrinsic language words from transliterable words is a key step aiding text processing tasks involving different natural languages. We consider the problem of unsupervised separation of transliterable words from native words for text in Malayalam language. Outlining a key observation on the diversity of characters beyond the word stem, we develop an optimization method to score words based on their nativeness. Our method relies on the usage of probability distributions over character n-grams that are refined in step with the nativeness scorings in an iterative optimization formulation. Using an empirical evaluation, we illustrate that our method, DTIM, provides significant improvements in nativeness scoring for Malayalam, establishing DTIM as the preferred method for the task.

1 Introduction

Malayalam is an agglutinative language from the southern Indian state of Kerala where it is the official state language. It is spoken by 38 million native speakers, three times as many speakers as Hungarian (Vincze et al., 2013) or Greek (Ntoulas et al., 2001), for which specialized techniques have been developed in other contexts. The growing web presence of Malayalam necessitates automatic techniques to process Malayalam text. A major hurdle in harnessing Malayalam text from social and web media for multi-lingual retrieval and machine translation is the presence of a large amount of transliterable words. By transliterable words, we mean both (a) words (from English) like *police* and *train* that virtually always appear in transliterated form in contemporary Malayalam, and (b) proper nouns such as names that need to be transliterated than translated to correlate with English text. On a manual analysis of a news article dataset, we found that transliterated words and proper nouns each form 10-12% of all distinct words. It is useful to transliterate such words for scenarios that involve processing Malayalam text in the company of English text; this will avoid them being treated as separate index terms (wrt their transliteration) in a multi-lingual retrieval engine, and help a statistical translation system to make use of the link to improve effectiveness. In this context, it ia notable that there has been recent interest in devising specialized methods to translate words that fall outside the core vocabulary (Tsvetkov and Dyer, 2015).

In this paper, we consider the problem of separating out such *transliterable* words from the other words within an unlabeled dataset; we refer to the latter as *"native"* words. We propose an *unsupervised* method, DTIM, that takes a dictionary of distinct words from a Malayalam corpus and scores each word based on their *nativeness*. Our optimization method, DTIM, iteratively refines the nativeness scoring of each word, leveraging dictionary-level statistics modelled using character n-gram probability distributions. Our empirical analysis establishes the effectiveness of DTIM.

We outline related work in the area in Section 2. This is followed by the problem statement in Section 3 and the description of our proposed approach in Section 4. Our empirical analysis forms Section 5 followed by conclusions in Section 7.

2 Related Work

Identification of transliterable text fragments, being a critical task for cross-lingual text analysis, has attracted attention since the 1990s. While most methods addressing the problem have used supervised learning, there have been some methods that can work without labeled data. We briefly survey both classes of methods.

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 155–164

2.1 Supervised and 'pseudo-supervised' Methods

An early work(Chen and Lee, 1996) focuses on a sub-problem, that of supervised identification of proper nouns for Chinese. (Jeong et al., 1999) consider leveraging decision trees to address the related problem of learning transliteration and back-transliteration rules for English/Korean word pairs. Recognizing the costs of procuring training data, (Baker and Brew, 2008) and (Goldberg and Elhadad, 2008) explore usage of pseudo-transliterable words generated using transliteration rules on an English dictionary for Korean and Hebrew respectively. Such pseudo-supervision, however, would not be able to generate uncommon domain-specific terms such as medical/scientific terminology for usage in such domains (unless specifically tuned), and is hence limited in utility.

2.2 Unsupervised Methods

A recent work proposes that multi-word phrases in Malayalam text where their component words exhibit strong co-occurrence be categorized as transliterable phrases (Prasad et al., 2014). Their intuition stems from observing contiguous words such as *test dose* which often occur in transliterated form while occurring together, but get replaced by native words in other contexts. Their method is however *unable to identify single transliterable words*, or phrases involving words such as *train* and *police* whose transliterations are heavily used in the company of native Malayalam words. A recent method for Korean (Koo, 2015) starts by identifying a seed set of transliterable words as those that begin or end with consonant clusters and have vowel insertions; this is specific to Korean since Korean words apparently do not begin or end with consonant clusters. High-frequency words are then used as seed words for native Korean for usage in a Naive Bayes classifier. In addition to the outlined reasons that make both the unsupervised methods inapplicable for our task, they both presume availability of corpus frequency statistics. We focus on a general scenario assuming the availability of only a word lexicon.

2.3 Positioning the Transliterable Word Identification Task

Nativeness scoring of words may be seen as a vocabulary stratification step (upon usage of thresholds) for usage by downstream applications. A multi-lingual text mining application that uses Malayalam and English text would benefit by transliterating non-native Malayalam words to English, so the transliterable Malayalam token and its transliteration is treated as the same token. For machine translation, transliterable words may be channeled to specialized translation methods (e.g., (Tsvetkov and Dyer, 2015)) or for manual screening and translation.

3 Problem Definition

We now define the problem more formally. Consider n distinct words obtained from Malayalam text, $\mathcal{W} = \{\ldots, w, \ldots\}$. Our task is to devise a technique that can use $\mathcal{W}$ to arrive at a *nativeness* score for each word, w, within it, as w_n. We would like w_n to be an accurate quantification of native-ness of word w. For example, when words in $\mathcal{W}$ are ordered in the decreasing order of w_n scores, we expect to get the native words at the beginning of the ordering and vice versa. We do not presume availability of any data other than $\mathcal{W}$; this makes our method applicable across scenarios where corpus statistics are unavailable due to privacy or other reasons.

3.1 Evaluation

Given that it is easier for humans to crisply classify each word as either native or transliterable (nouns or transliterated english words) in lieu of attaching a score to each word, the nativeness scoring (as generated by a scoring method such as ours) often needs to be evaluated against a crisp nativeness assessment, i.e., a scoring with scores in $\{0, 1\}$. To aid this, we consider the ordering of words in the labeled set in the decreasing (or more precisely, non-increasing) order of *nativeness* scores (each method produces an ordering for the dataset). To evaluate this ordering, we use two sets of metrics for evaluation:

- *Precision at the ends of the ordering:* **Top-k precision** denotes the fraction of *native* words within the k words at the *top* of the ordering; analogously, **Bottom-k precision** is the fraction of *transliterable* words among the *bottom* k. Since a good scoring would likely put native words at the top of the ordering and the transliterable ones at the bottom, a good scoring method would intuitively score

high on both these metrics. We call the average of the top-k and bottom-k precision for a given k, as **Avg-k precision**. These measures, evaluated at varying values of k, indicate the quality of the nativeness scoring.

- *Clustering Quality:* Consider the cardinalities of the native and transliterable words from the labeled set as being N and T respectively. We now take the top-N words and bottom-T words from the ordering generated by each method, and compare against the respective labeled sets as in the case of standard clustering quality evaluation[1]. Since the cardinalities of the generated native (transliterable) cluster and the native (transliterable) labeled set is both N (T), the **Recall** of the cluster is identical to its **Purity/Precision**, and thus, the **F-measure** too; we simply call it **Clustering Quality**. A cardinality-weighted average of the clustering quality across the native and transliterable clusters yields a single value for the clustering quality across the dataset. It may be noted that we are not making the labeled dataset available to the method generating the ordering, instead merely using it's cardinalities for evaluation purposes.

4 Our Method: DTIM

We now introduce our method, **D**iversity-based **T**ransliterable Word **I**dentification for **M**alayalam (DTIM). We use probability distributions over character n-grams to separately model transliterable and native words, and develop an optimization framework that alternatively refines the n-gram distributions and nativeness scoring within each iteration. DTIM involves an initialization that induces a "coarse" separation between native and transliterable words followed by iterative refinement. The initialization is critical in optimization methods that are vulnerable to local optima; the pure word distribution needs to be initialized to "coarsely" prefer pure words over transliterable words. This will enable further iterations to exploit the initial preference direction to further refine the model to "attract" the pure words more strongly and weaken any initial preference to transliterable words. The vice versa holds for the transliterable word models. We will first outline the initialization step followed by the description of the method.

4.1 Diversity-based Initialization

Our initialization is inspired by an observation on the variety of suffixes attached to a word stem. Consider a word stem |*pu*|*ra*|[2], a stem commonly leading to native Malayalam words; its suffixes are observed to start with a variety of characters such as |*ttha*| (e.g., |*pu*|*ra*|*ttha*|*kki*|), |*me*| (e.g., |*pu*|*ra*|*me*|), |*mbo*| (e.g., |*pu*|*ra*|*mbo*|*kku*|) and |*ppa*| (e.g., |*pu*|*ra*|*ppa*|*du*|). On the other hand, stems that mostly lead to transliterable words often do not exhibit so much of diversity. For example, |*re*|*so*| is followed only by |*rt*| (i.e., *resort*) and |*po*|*li*| is usually only followed by |*s*| (i.e., *police*). Some stems such as |*o*|*ppa*| lead to transliterations of two English words such as *open* and *operation*. Our observation, upon which we model the initialization, is that the variety of suffixes is generally correlated with native-ness (i.e., propensity to lead to a native word) of word stems. This is intuitive since non-native word stems provide limited flexibility to being modified by derivational or inflectional suffixes as compared to native ones.

For simplicity, we use the first two characters of each word as the word stem; we will evaluate the robustness of DTIM to varying stem lengths in our empirical evaluation, while consistently using the stem length of two characters in our description. We start by associating each distinct word stem in $\mathcal{W}$ with the number of unique third characters that follow it (among words in $\mathcal{W}$); in our examples, |*pu*|*ra*| and |*o*|*pa*| would be associated with 4 and 2 respectively. We initialize the *native-ness* weights as proportional to the diversity of 3^{rd} characters beyond the stem:

$$w_{n_0} = min\left\{0.99, \frac{|u3(w_{stem}, \mathcal{W})|}{\tau}\right\} \quad (1)$$

where $u3(w_{stem}, \mathcal{W})$ denotes the set of third characters that follow the stem of word w among words in $\mathcal{W}$. We flatten off w_{n_0} scores beyond a diversity of τ (note that a diversity of τ or higher will lead to the second term becoming 1.0 or higher, kicking in the min function to choose 0.99 for w_{n_0}) as shown in the above equation.

[1] https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html

[2] We will represent Malayalam words in transliterated form for reading by those who might not be able to read Malayalam. A pipe would separate Malayalam characters; for example |pu| corresponds to a single Malayalam character.

We give a small transliterable-ness weight even to highly diverse stems to reduce over-reliance on the initialization. We set $\tau = 10$ based on our observation from the dataset that most word stems having more than 10 distinct characters were seen to be native. As in the case of word stem length, we study DTIM trends across varying τ in our empirical analysis. w_{n_0} is in $[0, 1]$; analogously, $(1 - w_{n_0})$ may be regarded as a score of transliterableness.

4.2 Objective Function and Optimization Framework

As outlined earlier, we use separate character n-gram probability distributions to model native and transliterable words. We would like these probability distributions support the nativeness scoring, and vice versa. While the size of the n-grams (i.e., whether $n = 1, 2, 3$ or 4) is a system-level parameter, we use $n = 1$ for simplicity in our description. We denote the native and transliterable distributions as $\mathcal{N}$ and $\mathcal{T}$ respectively, with $\mathcal{N}(c)$ and $\mathcal{T}(c)$ denoting the weight associated with the character c according to the distributions. Consider the following function, given a particular state for the $\mathcal{N}$, $\mathcal{T}$ and w_ns:

$$\prod_{w \in \mathcal{W}} \prod_{c \in w} \left(w_n^2 \times \mathcal{N}(c) + (1 - w_n)^2 \times \mathcal{T}(c) \right) \quad (2)$$

This measures the aggregate supports for words in $\mathcal{W}$, the support for each word measured as an interpolated support from across the distributions $\mathcal{N}$ and $\mathcal{T}$ with weighting factors squares of the nativeness scores (i.e., w_ns) and transliterableness scores (i.e., $(1 - w_n)$s) respectively. Similar mixing models have been used earlier in emotion lexicon learning (Bandhakavi et al., 2014) and solution post discovery (Deepak and Visweswariah, 2014). The squares of the nativeness scores are used in our model (instead of the raw scores) for optimization convenience. A highly native word should intuively have a high w_n (nativeness) and a high support from $\mathcal{N}$ and correspondingly low transliterable-ness (i.e., $(1 - w_n)$) and support from $\mathcal{T}$; a highly transliterable word would be expected to have exactly the opposite. Due to the design of Eq. 2 in having the higher terms multiplied with each other (and so for the lower terms), this function would be maximized for a desirable estimate of the variables $\theta = \{\mathcal{N}, \mathcal{T}, \{\ldots, w_n, \ldots\}\}$. Conversely, by striving to maximize the objective function, we would arrive at a desirable estimate of the variables. An alternative construction yielding a minimizing objective would be as follows:

$$\prod_{w \in \mathcal{W}} \prod_{c \in w} \left((1 - w_n)^2 \times \mathcal{N}(c) + w_n^2 \times \mathcal{T}(c) \right) \quad (3)$$

In this form, given a good estimate of the variables, the native (transliterable) words have their nativeness (transliterableness) weights multiplied with the support from the transliterable (native) models. In other words, maximizing the objective in Eq. 2 is semantically similar to minimizing the objective in Eq. 3. As we will illustrate soon, it is easier to optimize for $\mathcal{N}$ and $\mathcal{T}$ using the maximizing formulation in Eq. 2 while the minimizing objective in Eq. 3 yields better to optimize for the word nativeness scores, $\{\ldots, w_n, \ldots\}$.

4.3 Learning $\mathcal{N}$ and $\mathcal{T}$ using the Maximizing Objective

We start by taking the log-form of the objective in Eq. 2 (this does not affect the optimization direction), yielding:

$$\mathcal{O}_{max} = \sum_{w \in \mathcal{W}} \sum_{c \in w} ln \left(w_n^2 \times \mathcal{N}(c) + (1 - w_n)^2 \times \mathcal{T}(c) \right) \quad (4)$$

The distributions, being probability distributions over n-grams, should sum to zero. This constraint, for our unigram models, can be written as:

$$\sum_c \mathcal{N}(c) = \sum_c \mathcal{T}(c) = 1 \quad (5)$$

Fixing the values of $\{\ldots, w_n, \ldots\}$ and $\mathcal{T}$ (or $\mathcal{N}$), we can now identify a better estimate for $\mathcal{N}$ (or $\mathcal{T}$) by looking for an optima (i.e., where the objective function has a slope of zero). Towards that, we take the partial derivative (or slope) of the objective for a particular character. :

$$\frac{\partial \mathcal{O}_{max}}{\partial \mathcal{N}(c')} = \left(\sum_{w \in \mathcal{W}} \frac{freq(c', w) \times w_n^2}{\left(w_n^2 \mathcal{N}(c') + (1 - w_n)^2 \mathcal{T}(c') \right)} \right) + \lambda_{\mathcal{N}} \quad (6)$$

where $freq(c', w)$ is the frequency of the character c' in w and $\lambda_{\mathcal{N}}$ denotes the Lagrangian multiplier corresponding to the sum-to-unity constraint for $\mathcal{N}$. Equating this to zero does not however

yield a closed form solution for $\mathcal{N}'$, but a simple re-arrangement yields an iterative update formula:

$$\mathcal{N}(c') \propto \sum_{w \in \mathcal{W}} \frac{freq(c', w) \times w_n^2}{\left(w_n^2 + (1 - w_n)^2 \frac{\mathcal{T}(c')}{\mathcal{N}_P(c')}\right)} \quad (7)$$

The $\mathcal{N}$ term in the RHS is denoted as $\mathcal{N}_P$ to indicate the usage of the previous estimate of $\mathcal{N}$. The sum-to-one constraint is trivially achieved by first estimating the $\mathcal{N}(c')$s by treating Eq. 7 as equality, followed by normalizing the scores across the character vocabulary. Eq. 7 is intuitively reasonable, due to establishing a somewhat direct relationship between $\mathcal{N}$ and w_n (in the numerator), thus allowing highly native words to contribute more to building $\mathcal{N}$. The analogous update formula for $\mathcal{T}$ fixing $\mathcal{N}$ turns out to be:

$$\mathcal{T}(c') \propto \sum_{w \in \mathcal{W}} \frac{freq(c', w) \times (1 - w_n)^2}{\left((1 - w_n)^2 + w_n^2 \frac{\mathcal{N}(c')}{\mathcal{T}_P(c')}\right)} \quad (8)$$

Eq. 7 and Eq. 8 would lead us closer to a maxima for Eq. 4 is their second (partial) derivatives are negative[3]. To verify this, we note that the second (partial) derivative wrt $\mathcal{N}(c')$ is as follows

$$\frac{\partial^2 \mathcal{O}_{max}}{\partial^2 \mathcal{N}(c')} =$$

$$(-1) \times \sum_{w \in \mathcal{W}} \frac{freq(c', w)(w_n^2)^2}{\left(w_n^2 \mathcal{N}(c') + (1 - w_n)^2 \mathcal{T}(c')\right)^2} \quad (9)$$

It is easy to observe that the RHS is a product of -1 and a sum of a plurality of positive terms (square terms that are trivially positive, with the exception being the $freq(.,.)$ term which is also non-negative by definition), altogether yielding a negative value. That the the second (partial) derivative is negative confirms that the update formula derived from the first partial derivative indeed helps in maximizing $\mathcal{O}_{max}$ wrt $\mathcal{N}(c')$. A similar argument holds for the $\mathcal{T}(c')$ updates as well, which we omit for brevity.

[3] http://mathworld.wolfram.com/SecondDerivativeTest.html

4.4 Learning the nativeness scores, $\{\ldots, w_n, \ldots\}$, using the Minimizing Objective

Analogous to the previous section, we take the log-form of Eq. 3:

$$\mathcal{O}_{min} = \sum_{w \in \mathcal{W}} \sum_{c \in w} ln \left((1 - w_n)^2 \times \mathcal{N}(c) + w_n^2 \times \mathcal{T}(c) \right) \quad (10)$$

Unlike the earlier case, we do not have any constraints since the sum-to-unit constraint on the nativeness and transliterableness scores are built in into the construction. We now fix the values of all other variables and find the slope wrt w'_n, where w' indicates a particular word in $\mathcal{W}$.

$$\frac{\partial \mathcal{O}_{min}}{\partial w'_n} = \sum_{c \in w'} \frac{2w'_n \mathcal{T}(c) + 2w'_n \mathcal{N}(c) - 2\mathcal{N}(c)}{\left(w'^2_n \mathcal{T}(c) + (1 - w'_n)^2 \mathcal{N}(c)\right)} \quad (11)$$

We equate the slope to zero and form an iterative update formula, much like in the distribution estimation phase.

$$w'_n = \frac{\sum_{c \in w'} \frac{\mathcal{N}(c)}{(w'^2_n \mathcal{T}(c) + (1 - w'_n)^2 \mathcal{N}(c))}}{\sum_{c \in w'} \frac{\mathcal{N}(c) + \mathcal{T}(c)}{(w'^2_n \mathcal{T}(c) + (1 - w'_n)^2 \mathcal{N}(c))}} \quad (12)$$

Using the previous estimates of w'_n for the RHS yields an iterative update form for the nativeness scores. As in the model estimation phase, the update rule establishes a reasonably direct relationship between w'_n and $\mathcal{N}$. Since our objective is to minimize $\mathcal{O}_{min}$, we would like to verify the direction of optimization using the second partial derivative.

$$\frac{\partial^2 \mathcal{O}_{min}}{\partial^2 w'_n} =$$

$$\sum_{c \in w'} \frac{\mathcal{N}(c)\mathcal{T}(c) - \left(w'_n \mathcal{T}(c) - (1 - w'_n)\mathcal{N}(c)\right)^2}{\left(w'^2_n \mathcal{T}(c) + (1 - w'_n)^2 \mathcal{N}(c)\right)^2} \quad (13)$$

We provide an informal argument for the positivity of the second derivative; note that the denominator is a square term making it enough to analyze just the numerator term. Consider a highly native word (high w'_n) whose characters would intuitively satisfy $\mathcal{N}(c) > \mathcal{T}(c)$. For the boundary case of $w'_n = 1$, the numerator term reduces

to $\mathcal{T}(c) \times (\mathcal{N}(c) - \mathcal{T}(c))$ which would be positive given the expected relation between $\mathcal{N}(c)$ and $\mathcal{T}(c)$. A similar argument holds for highly transliterable words. For words with $w'_n \to 0.5$ where we would expect $\mathcal{N}(c) \approx \mathcal{T}(c)$, the numerator becomes $\mathcal{N}(c)\mathcal{T}(c) - 0.25(\mathcal{T}(c) - \mathcal{N}(c))^2$, which is expected to be positive since the difference term is small, making it's square very small in comparison to the first product term. To outline the informal nature of the argument, it may be noted that $\mathcal{T}(c) > \mathcal{N}(c)$ may hold for certain characters within highly native words; but as long as most of the characters within highly native words satisfy the $\mathcal{N}(c) > \mathcal{T}(c)$, there would be sufficient positivity to offset the negative terms induced with such outlier characters.

Algorithm 1: DTIM

Input: A set of Malayalam words, $\mathcal{W}$
Output: A nativeness scoring $w_n \in [0, 1]$ for every word w in $\mathcal{W}$
Hyper-parameters: word stem length, τ, n
Initialize the w_n scores for each word using the diversity metric in Eq. 1 using the chosen stem length and τ
while *not converged and number of iterations not reached* **do**
  Estimate n-gram distributions $\mathcal{N}$ and $\mathcal{T}$ using Eq. 7 and Eq. 8 respectively
  Learn nativeness weights for each word using Eq. 12
end
return *latest estimates of nativeness weights*

4.5 DTIM: The Method

Having outlined the learning steps, the method is simply an iterative usage of the learning steps as outlined in Algorithm 1. In the first invocation of the distribution learning step where previous estimates are not available, we simply assume a uniform distribution across the n-gram vocabulary for usage as the previous estimates. Each of the update steps are linear in the size of the dictionary, making DTIM a computationally lightweight method. Choosing $n = 2$ instead of unigrams (as used in our narrative) is easy since that simply involves replacing the $c \in w$ all across the update steps by $[c_i, c_{i+1}] \in w$ with $[c_i, c_{i+1}]$ denoting pairs of contiguous characters within the word; similarly, $n = 3$ involves usage of contiguous character triplets and correspondingly learning the distributions $\mathcal{N}$ and $\mathcal{T}$ over triplets. The DTIM structure is evidently inspired by the Expectation-Maximization framework (Dempster et al., 1977) involving alternating optimizations of an objective function; DTIM, however, uses different objective functions for the two steps for optimization convenience.

5 Experiments

We now describe our empirical study of DTIM, starting with the dataset and experimental setup leading on to the results and analyses.

5.1 Dataset

We evaluate DTIM on a set of 65068 distinct words from across news articles sourced from *Mathrubhumi*[4], a popular Malayalam newspaper; this word list is made available publicly[5]. For evaluation purposes, we got a random subset of 1035 words labeled by one of three human annotators; that has been made available publicly[6] too, each word labeled as either *native*, *transliterable* or *unknown*. There were approximately 3 native words for every transliterable word in the labeled set, reflective of distribution in contemporary Malayalam usage as alluded to in the introduction. We will use the whole set of 65068 words as input to the method, while the evaluation would obviously be limited to the labelled subset of 1035 words.

5.2 Baselines

As outlined in Section 2, the unsupervised version of the problem of telling apart native and transliterable words for Malayalam and/or similar languages has not been addressed in literature, to the best of our knowledge. The unsupervised Malayalam-focused method(Prasad et al., 2014) (Ref: Sec 2.2) is able to identify only transliterable word-pairs, making it inapplicable for contexts such as our health data scenario where individual english words are often transliterated for want of a suitable malayalam alternative. The Korean method(Koo, 2015) is too specific to Korean language and cannot be used for other languages due to the absence of a generic high-precision rule to identify a seed set of transliterable words. With both the unsupervised state-of-the-art approaches being inapplicable for our task,

[4] http://www.mathrubhumi.com
[5] Dataset: https://goo.gl/DOsFES
[6] Labeled Set: https://goo.gl/XEVLWv

we compare against an intuitive generalization-based baseline, called **GEN**, that orders words based on their support from the combination of a unigram and bi-gram character language model learnt over $\mathcal{W}$; this leads to a scoring as follows:

$$w_n^{GEN} = \prod_{[c_i,c_{i+1}]\in w} \lambda \times B_{\mathcal{W}}(c_{i+1}|c_i) + (1-\lambda) \times U_{\mathcal{W}}(c_{i+1}) \quad (14)$$

where $B_{\mathcal{W}}$ and $U_{\mathcal{W}}$ are bigram and unigram character-level language models built over all words in $\mathcal{W}$. We set $\lambda = 0.8$ (Smucker and Allan, 2006). We experimented with higher-order models in **GEN**, but observed drops in evaluation measures leading to us sticking to the usage of the unigram and bi-gram models. The form of Eq. 14 is inspired by an assumption similar to that used in both (Prasad et al., 2014) and (Koo, 2015) that transliterable words are rare. Thus, we expect they would not be adequately supported by models that generalize over whole of $\mathcal{W}$. We also compare against our diversity-based initialization score from Section 4.1, which we will call as **INIT**. For ease of reference, we outline the **INIT** scoring:

$$w_n^{INIT} = min\left\{0.99, \frac{|u3(w_{stem}, \mathcal{W})|}{\tau}\right\} \quad (15)$$

The comparison against **INIT** enables us to isolate and highlight the value of the iterative update formulation vis-a-vis the initialization.

5.3 Evaluation Measures and Setup

As outlined in Section 3, we use *top-k*, *bottom-k* and *avg-k* precision (evaluated at varying values of k) as well as *clustering quality* in our evaluation. For the comparative evaluaton, we set DTIM parameters as the following: $\tau = 10$ and a word-stem length of 2. We will study trends against variations across these parameters in a separate section.

5.4 Experimental Results

5.4.1 Precision at the ends of the Ordering

Table 1 lists the precision measures over various values of k. It may be noted that any instantiation of DTIM (across the four values of n-gram size, n) is able to beat the baselines convincingly on each metric on each value of k, convincingly establishing the effectiveness of the DTIM formulation. DTIM is seen to be much more effective in separating out the native and transliterable words at either ends of the ordering, than the baselines. It is also notable that EM-style iterations are able to significantly improve upon the initialization (i.e., INIT). That the bottom-k precision is seen to be consistently lower than top-k precision needs to be juxtaposed in the context of the observation that there were only 25% transliterable words against 75% native words; thus, the lift in precision against a random ordering is much more substantial for the transliterable words. The trends across varying n-gram sizes (i.e., n) in DTIM is worth noting too; the higher values of n (such as 3 and 4) are seen to make more errors at the ends of the lists, whereas they catch-up with the $n \in \{1, 2\}$ versions as k increases. This indicates that smaller-n DTIM is being able to tell apart a minority of the words exceedingly well (wrt native-ness), whereas the higher n-gram modelling is able to spread out the gains across a larger spectrum of words in $\mathcal{W}$. Around $n = 4$ and beyond, sparsity effects (since 4-grams and 5-grams would not occur frequently, making it harder to exploit their occurrence statistics) are seen to kick in, causing reductions in precision.

5.4.2 Clustering Quality

Table 2 lists the clustering quality metric across the various methods. Clustering quality, unlike the precision metrics, is designed to evaluate the entire ordering without limiting the analysis to just the top-k and bottom-k entries. As in the earlier case, DTIM convincingly outperforms the baselines by healthy margins across all values of n. Consequent to the trends across n as observed earlier, $n \in \{3, 4\}$ are seen to deliver better accuracies, with such gains tapering off beyond $n = 4$ due to sparsity effects. The words, along with the DTIM nativeness scores for $n = 3$, can be viewed at `https://goo.gl/OmhlB3` (the list excludes words with fewer than 3 characters).

5.5 Analyzing DTIM

We now analyze the performance of DTIM across varying values of the diversity threshold (τ) and word-stem lengths.

	k=50			k=100			k=150			k=200		
	Top-k	Bot-k	Avg-k	Top-k	Bot-k	Avg-k	Top-k	Bot-k	Avg-k	Top-k	Bot-k	Avg-k
INIT	0.88	0.50	0.69	0.90	0.40	0.65	0.90	0.41	0.66	0.90	0.38	0.64
GEN	0.64	0.10	0.37	0.58	0.11	0.35	0.60	0.15	0.38	0.64	0.17	0.41
DTIM (n=1)	0.94	0.64	0.79	0.90	0.56	0.73	0.90	0.49	0.70	0.92	0.48	0.70
DTIM (n=2)	**1.00**	**0.78**	**0.89**	**0.94**	0.68	0.81	**0.93**	0.57	0.75	**0.95**	0.52	0.74
DTIM (n=3)	0.86	0.76	0.81	0.91	**0.75**	**0.83**	0.92	**0.69**	**0.81**	0.92	0.64	**0.78**
DTIM (n=4)	0.82	0.74	0.78	0.87	0.69	0.78	0.83	0.62	0.73	0.85	**0.65**	0.75

Table 1: Top-k and Bottom-k Precision (best result in each column highlighted)

	Native	Transliterable	Weighted Average
INIT	0.79	0.38	0.69
GEN	0.73	0.17	0.59
DTIM (n=1)	0.81	0.44	0.72
DTIM (n=2)	0.84	0.50	0.75
DTIM (n=3)	**0.86**	**0.60**	**0.79**
DTIM (n=4)	**0.86**	**0.60**	**0.79**

Table 2: Clustering Quality (best result in each column highlighted)

$\tau \rightarrow$	5	10	20	50	100	1000
$n = 1$	0.72	0.72	0.72	0.72	0.72	0.72
$n = 2$	0.74	0.75	0.75	0.74	0.74	0.74
$n = 3$	0.77	0.79	0.78	0.78	0.78	0.78
$n = 4$	0.78	0.79	0.79	0.79	0.79	0.79

Table 3: DTIM Clustering Quality against τ

Stem Length $\rightarrow$	1	2	3	4
$n = 1$	0.64	0.72	**0.75**	0.56
$n = 2$	0.58	**0.75**	0.74	0.55
$n = 2$	0.59	**0.79**	0.69	0.60
$n = 2$	0.58	**0.79**	0.69	0.62

Table 4: DTIM Clustering Quality against Word Stem Length (best result in each row highlighted)

5.5.1 Diversity Threshold

Table 3 analyzes the clustering quality trends of DTIM across varying values of τ. The table suggests that DTIM is extremely robust to variations in diversity threshold, despite a slight preference towards values around 10 and 20. This suggests that a system designer looking to use DTIM need not carefully tune this parameter due to the inherent robustness.

5.5.2 Word Stem Length

Given the nature of Malayalam language where the variations in word lengths are not as high as in English, it seemed very natural to use a word stem length of 2. Moreover, very large words are uncommon in Malayalam. In our corpus, 50% of words were found to contain five characters or less, the corresponding fraction being 71% for upto seven characters. Our analysis of DTIM across variations in word-stem length, illustrated in Table 4 strongly supports this intuition with clustering quality peaking at stem-length of 2 for $n \geq 2$. It is notable, however, that DTIM degrades gracefully on either side of 2. Trends across different settings of word-stem length are interesting since they may provide clues about applicability for other languages with varying character granularities (e.g., each Chinese character corresponds to multiple characters in Latin-script).

6 Discussion

6.1 Applicability to Other Languages

In contrast to earlier work focused on specific languages (e.g., (Koo, 2015)) that use heuristics that are very specific to the language (such as expected patterns of consonants), DTIM heuristics are general-purpose in design. The only heuristic setting that is likely to require some tuning for applicability in other languages is the word-stem length. We expect the approach would generalize

well to other Sanskrit-influenced Dravidian languages such as Kannada/Telugu. Unfortunately, we did not have any Kannada/Telugu/Kodava knowledge (Dravidian languages have largely disjoint speaker-populations) in our team, or access to labelled datasets in those languages (they are low-resource languages too); testing this on Kannada/Telugu/Tamil would be interesting future work. The method is expected to be less applicable to English, the language being significantly different and with potentially fewer transliterable words.

6.2 DTIM in an Application Context

Within any target application context, machine-labelled transliterable words (and their automatically generated transliterations) may need to manual screening for accountability reasons. The high accuracy at either ends of the ordering lends itself to be exploited in the following fashion; in lieu of employing experts to verify all labellings/transliterations, low-expertise volunteers (e.g., students) can be called in to verify labellings at the ends (top/bottom) of the lists with experts focusing on the middle (more ambiguous) part of the list; this frees up experts' time as against a cross-spectrum expert-verification process, leading to direct cost savings. We also expect that DTIM followed by automatic transliterations of bottom-k words would aid in retrieval and machine translation scenarios.

7 Conclusions and Future Work

We considered the problem of unsupervised separation of transliterable and native words in Malayalam, a critical task in easing automated processing of Malayalam text in the company of other language text. We outlined a key observation on the differential diversity beyond word stems, and formulated an initialization heuristic that would coarsely separate native and transliterable words. We proposed the usage of probability distributions over character n-grams as a way of separately modelling native and transliterable words. We then formulated an iterative optimization method that alternatively refines the nativeness scorings and probability distributions. Our technique for the problem, DTIM, that encompasses the initialization and iterative refinement, was seen to significantly outperform other unsupervised baseline methods in our empirical study. This establishes DTIM as the preferred method for the task. We have also released our datasets and labeled subset to help aid future research on this and related tasks.

7.1 Future Work

The applicability of DTIM to other Dravidian languages is interesting to study. Due to our lack of familiarity with any other language in the family, we look forward to contacting other groups to further the generalizability study. While nativeness scoring improvements directly translate to reduction of effort for manual downstream processing, quantifying gains these bring about in translation and retrieval is interesting future work. Exploring the relationship/synergy of this task and Sandhi splitting (Natarajan and Charniak, 2011) would form another interesting direction for future work. Further, we would like to develop methods to separate out the two categories of transliterable words, viz., foreign language words and proper nouns. Such a method would enable a more fine-grained stratification of the vocabulary.

Transliterable words are often within Malayalam used to refer to very topical content, for which suitable words are harder to find. Thus, transliterable words could be preferentially treated towards building rules in interpretable clustering (Balachandran et al., 2012) and for modelling context in regex-oriented rule-based information extraction (Murthy et al., 2012). Transliterable words might also hold cues for detecting segment boundaries in conversational transcripts (Kummamuru et al., 2009; Padmanabhan and Kummamuru, 2007).

References

[Baker and Brew2008] Kirk Baker and Chris Brew. 2008. Statistical identification of english loanwords in korean using automatically generated training data. In *LREC*. Citeseer.

[Balachandran et al.2012] Vipin Balachandran, P Deepak, and Deepak Khemani. 2012. Interpretable and reconfigurable clustering of document datasets by deriving word-based rules. *Knowledge and information systems*, 32(3):475–503.

[Bandhakavi et al.2014] Anil Bandhakavi, Nirmalie Wiratunga, P Deepak, and Stewart Massie. 2014. Generating a word-emotion lexicon from# emotional tweets. In ** SEM@ COLING*, pages 12–21.

[Chen and Lee1996] Hsin-Hsi Chen and Jen-Chang

Lee. 1996. Identification and classification of proper nouns in chinese texts. In *COLING*.

[Deepak and Visweswariah2014] P Deepak and Karthik Visweswariah. 2014. Unsupervised solution post identification from discussion forums. In *ACL (1)*, pages 155–164.

[Dempster et al.1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

[Goldberg and Elhadad2008] Yoav Goldberg and Michael Elhadad. 2008. Identification of transliterated foreign words in hebrew script. In *Computational linguistics and intelligent text processing*, pages 466–477. Springer.

[Jeong et al.1999] Kil Soon Jeong, Sung Hyon Myaeng, Jae Sung Leeb, and Key-Sun Choib. 1999. Automatic identification and back-transliteration of foreign words for information retrieval. *Inf. Proc. & Mgmt.*, 35.

[Koo2015] Hahn Koo. 2015. An unsupervised method for identifying loanwords in korean. *Language Resources and Evaluation*, 49(2):355–373.

[Kummamuru et al.2009] Krishna Kummamuru, Deepak Padmanabhan, Shourya Roy, and L Venkata Subramaniam. 2009. Unsupervised segmentation of conversational transcripts. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2(4):231–245.

[Murthy et al.2012] Karin Murthy, P Deepak, and Prasad M Deshpande. 2012. Improving recall of regular expressions for information extraction. In *International Conference on Web Information Systems Engineering*, pages 455–467. Springer.

[Natarajan and Charniak2011] Abhiram Natarajan and Eugene Charniak. 2011. s^3 - statistical sandhi splitting. In *IJCNLP*, pages 301–308.

[Ntoulas et al.2001] Alexandros Ntoulas, Sofia Stamou, and Manolis Tzagarakis. 2001. Using a www search engine to evaluate normalization performance for a highly inflectional language. In *ACL (Companion Volume)*, pages 31–36.

[Padmanabhan and Kummamuru2007] Deepak Padmanabhan and Krishna Kummamuru. 2007. Mining conversational text for procedures with applications in contact centers. *International Journal on Document Analysis and Recognition*, 10(3):227–238.

[Prasad et al.2014] Reshma Prasad, Mary Priya Sebastian, et al. 2014. A technique to extract transliterating phrases in statistical machine translation from english to malayalam. In *National Conference on Indian Language Computing*.

[Smucker and Allan2006] Mark Smucker and James Allan. 2006. An investigation of dirichlet prior smoothing's performance advantage. Ir.

[Tsvetkov and Dyer2015] Yulia Tsvetkov and Chris Dyer. 2015. Lexicon stratification for translating out-of-vocabulary words. In *ACL Short Papers*, pages 125–131.

[Vincze et al.2013] Veronika Vincze, János Zsibrita, and István Nagy. 2013. Dependency parsing for identifying hungarian light verb constructions. In *IJCNLP*, pages 207–215.

Known Strangers:
Cross Linguistic Patterns in Multilingual Multidirectional Dictionaries

Dr. Rejitha K. S.
Linguistic Data Consortium
Central Institute of Indian Languages
Manasa Gangothri, Mysore.
ksrejitha@gmail.com

Rajesha N.
Linguistic Data Consortium
Central Institute of Indian Languages
Manasa Gangothri, Mysore
n.rajesha@yahoo.co.in

Abstract

The multilingual multidirectional dictionary gives the linguistic equivalent across the languages. In order to build such a dictionary in electronic form poses considerable challenges to the lexicographer and the dictionary architect. One of the major challenges is linking lexical ambiguity across languages. This paper intends to address that issue along with many other challenges involved in creating such a multilingual and multidirectional dictionary.

1 Introduction:

Dictionaries are compiled with a view to provide lexical and semantic information from thousands of years. Electronic/digital dictionary does the same by replacing the format of the traditional printed dictionaries. An electronic dictionary, though primarily designed to provide basic information such as grammatical category, meaning, usage etc. as the paper dictionaries, they can also provide additional information like pronunciation, motion pictures through multimedia which paper dictionaries cannot.

The expression Electronic dictionary gained momentum in the last quarter of the 20th century as a term for a specialized device - a handheld computer dedicated to storing a lexical database and performing lookup in it. Classical lexicography demands a complex relationship with linguistic theory. So is electronic lexicography with computational linguistics. Electronic dictionaries are a product of this association and they also serve as tools and feedstock for creating other products. An electronic bilingual or multilingual dictionary may be a digitized edition of a conventional reference work perhaps augmented by types of information specific of this medium (recorded pronunciations, hyperlinks, full text search etc.). Alternatively, it may be a system of monolingual dictionaries of different languages interlinked at the level of entries. [Ivan A Derhanski 2009]

If the construction of the multilingual electronic dictionary is not just a collection of digitized versions of printed dictionaries but to offer facilities like multidirectional search, extracting mono-lingual, bi-lingual, tri-lingual dictionaries, root lexicons and even provide backend support for translation systems then designing such a dictionary database throws practical challenges. Especially when such database accommodates multiple languages at one go and provides options for multidirectional search. That means word of any language as source can be sought in one or more target languages catered by the dictionary system.

The creation of a multilingual dictionary database concerns itself with the source of information used for constructing them. Most of such endeavors primarily rely on printed dictionaries or machine readable versions of the same. Currently we have the advantage of electronic corpora which has been built for many Indian Languages over the past decade.

Polysemy is seldom a serious problem in human communication. Lexicographers have traditionally been concerned with the best way to account for the fact that one word can carry several different meanings (Leacock C. and Ravin 2000). Over time, lexicographic procedures have been established that have resulted in the listing of multiple dictionary senses for polysemous words where sub-senses are grouped together with their respective definitions (Henri Béjoint 2000).

This paper addresses how the concepts described in a lingua-franca provides a basis for conducting cross-linguistic research there by facilitating the creation of multilingual dictionary capable of overcoming a number of important linguistic problems.

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 165–171

The lexical under-specifications and lexical ambiguity are among the major problems. Sometimes one leads to the other. Lexical ambiguity is one of the issues that a lexicographer and the dictionary architect have to face. This paper describes the observations that a lexicographer encounters while handling prototype of 'concept-set-model' architecture.

2 Review of literature:

When we took up the task of building multilingual, multidirectional dictionary for Indian languages, we researched few previous initiatives. The Universidad Politécnica de Madrid's School of Computing have developed a system for building multilingual dictionaries based on multiple term equivalences known as universal words. The system is based on Princeton University's WordNet database. WordNet is a lexical database developed by linguists at Princeton's Cognitive Science Laboratory. The database was designed to inventory, classify and relate the semantic and lexical content of the English language. The system's other mainstay are universal words. The concept of universal word came out of the UNL (Universal Networking Language) Project. The aim of this project is to eliminate the barriers of linguistic diversity by creating a medium of information exchange through which users can communicate in their own language. Similar attempts were done in PanLex Project that aims to help one to express any lexical concept in any language. The endeavor like BabelNet, which is developed with lexicographic and encyclopedic coverage of terms, is a semantic network which connects concepts and named entities in a very large network of semantic relations, called Babel synsets. Each Babel synset represents a given meaning and contains all the synonyms which express that meaning in a range of different languages.

3 Our Approach:

Since our objective is narrow and is to make a corpus based dictionary of Indian languages and not of a semantic net. The paper is about dictionary only. To make a multilingual multidirectional digital dictionary the approach of word to word linking across languages is not practically feasible. Some concepts may never have a word for it because the concept itself could be alien to the language culture. For example, there cannot be an equivalent word for Kannada '*muḍḍe*'. *muḍḍe* is a kind of edible ball prepared by cooking millet powder used majorly in southern part of Karnataka. So does for 'tulip' flower in Telugu. Since the tulip flower is not native to the culture of the Telugu speaking land. In traditional dictionaries, such cases are dealt with by describing source language word in target language.

Word from language 'A' may have more than one meaning which gets connected to a word in language 'B' which may not share all the meanings of the language 'A'. Sometimes it may have other meanings too which language 'A' word may not have.

Fixing a language as source and other languages as target may bring only the concepts of the source language culture and omits all possible concepts that other languages may have. A dictionary database based on such limited concepts offers limited descriptions to the end-user, primarily if the end user is searching between two languages which are only target languages in the database architecture. Making a universal word-set is a good start but it will eventually lack the language specific or region specific concepts in the multilingual multidirectional dictionary.

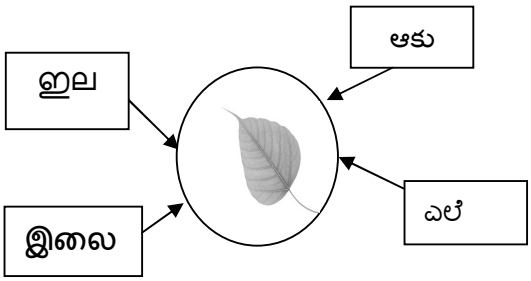
Words borrowed from same origin like Proto-Dravidian or Sanskrit to two different languages may not carry the same concept with them. So it is evident that word to word linking across languages is not a feasible solution even at the stage of polysemy or borrowed words like *tatsamas*. Thus we have to lean back to the basic principles of linguistics where it is the concept that exists as the fact and we label it differently in different languages.

As the Vedic hymn say "*Ékam sáth* víprā bahudhā́ *vadanti*". (The fact exists and the learned one call it by different names -*Rigveda*) The world existed before any language came into existence. When languages evolved with its vocabulary its primary job was to label the things and actions. Those words later fell into different grammatical categories like noun, verb, adjective, adverb etc.

According to Ferdinand de Saussure the signified is the concept, the meaning, the thing indicated by the signifier (Language). It need not be a 'real object' but is some referent to which the signifier refers. The language is built around the concepts that exist in environment.

Let us consider the concept 'leaf" and its description as 'The main organ of photosynthesis and transpiration in higher plants'[1]. This

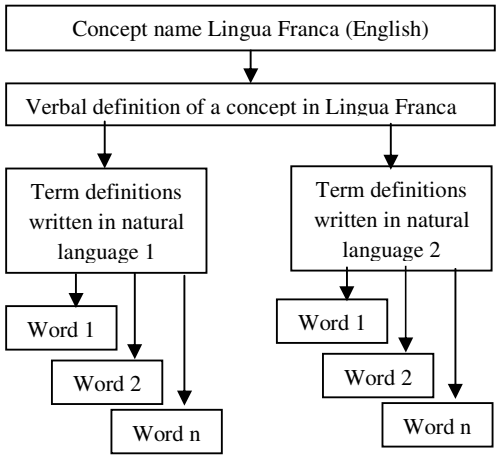
concept-set idea of leaf in four Dravidian languages Malayalam, Tamil, Telugu, and Kannada is as follows.



In Terminology, terms i.e. the "verbal definition of a concept" need to be separated from concept names since they belong to two different semiotic systems. The first is a linguistic system while the second is conceptual. Similarly, term definitions written in natural language need to be separated from concept definitions written in a formal language. The former are viewed as linguistic explanations while the latter are considered logical specifications of concept. The result is a new kind of terminology called onto terminology (Christophe Roche, Marie Calberg-Challot, Luc Damas, Philippe Rouard 2009)

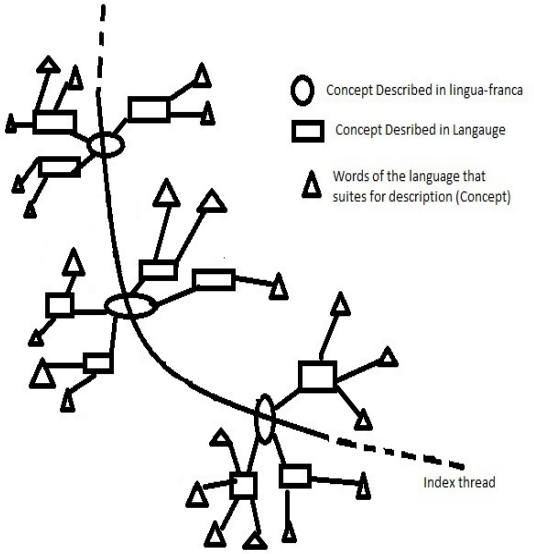
On the similar lines of onto terminology we build our concept set which is a basic data unit for a lexical entry. A concept set is a set which has a concept described in Lingua-franca along with its associated sense in connected languages which in turn connected to related words in Indian languages catered by the dictionary.

Our typical concept set looks as follows.



The 'concept-set-model' i.e. a Lexical item is entered along with its synonyms and semantic meaning linked with 'concept' (descriptive meaning in lingua-franca) into the database. We have chosen English as lingua-franca with its probable word if exists in English. Based on the concept the process is iterated in other languages. In other words, we are following indexation of 'concept'. Here word is terminal or leaf end of the linkage and not like a node of a semantic network model.

In Central Institute of Indian Languages (CIIL) the concept-set model based dictionary architecture was built in 2010 (Rajesha N, Ramya M and Samar Sinha 2011). We have the advantage of electronic corpora which was built in house for many Indian Languages over the past decade by CIIL. We thought of using the same to enrich our dictionary named '*vāgartha*' (word and sense) that we are building in-house.



Since we are following the concept which is the fact that exists, rather than any words to connect with, the challenge of choosing a fixed primary language is also eliminated. For the purpose of management and to avoid confusions, the method of entering the new concepts into the dictionary restricts to one language at a given point of time. Such language will be called as Primary Language. All other languages will add the entries and other respective fields in their language in correspondence with the concepts given by the Primary Language. After a fixed period a different language will become the Primary language so that the dictionary should not miss any concepts which could be a cultural specific item of a language community/region. We devised a system where the concept is fixed and the words act as labels attached to the concepts. The concept and the primary language word associated with it, is shown to the

connecting lexicographer. The system gives a facility to transliterate the primary language word if it would be of any help to the lexicographer.

4 Observations:

Looking beyond the well-known issues surrounding the treatment of polysemy in a single language we find even greater problems when it comes to accounting for polysemy across languages. Overcoming these problems is not only important for the design of traditional lexicons but also crucial for the successful implementation Multilingual Lexical Databases. (Hans Christian Boas 2009)

Polysemy can pose problems in intra-lingual and inter-lingual linkages.

4.1 Lexical Ambiguity in a language

In Intra-lingual linking the Lexical Ambiguity words that are not even remotely connected in conceptual sense bring ambiguity to the user. For example the Malayalam word 'ʋaɽʃam' has two senses as following.

1. 'Year- A period of time containing 365 (or 366) days'[1]
2. 'Rain- Water falling in drops from vapour condensed in the atmosphere'[1].

The Dictionary database architect has to arrange the data without any redundancy in relational database. So the single lexical entry of the word has to be connected with two or more senses here. None of them is a sub-sense of the other.

An end user search of database for Malayalam word '*aːŋʈə*' should fetch the description as well as the synonyms of '*aːŋʈə*'; in such a case it will obviously fetch '*vaɽʃam*'. But because the '*vaɽʃam*' is connected with other words (like '*maɻa*') in the sense of 'Rain', database should not render '*maɻa*' for '*aːŋʈə*'.

Tautologous: The organization of data should follow the guideline. i.e., words should be interlinked with all other synonyms and the concept to which it is related. While writing dictionary definitions many lexicographers follow precise guidelines on how to define a word.

In spite of this we find definitions like

Luncher — 'Someone who is eating lunch'[1]

Magnetism — 'The branch of science that studies magnetism'[1]

These definitions are logically sound and literally true but they are also tautologous. They use the same words or roots in the definition as are found in the headword. The lexicographer has to understand that the architecture of the database will be such that the definitions are not only for the headword but to all the synonyms to which the sense is connected. All these synonyms are also headword candidates and part of lexicon of that language. So none of those words should be used in definition which leading to tautologous entries.

4.2 Lexical Ambiguity across language

In practical scenario we observed four different types of cross linguistic patterns and two potentially confusing patterns. The following table gives a description of these observations in the multilingual database.

Patterns	Description
A = A	Complete overlapping of word senses
A ≠ A	No overlapping of word senses even if words belongs to the same origin or word conceptualization
A1 = A1 A2 ≠ A2	Semi overlapping of word senses. The word may be having more than one sense in a language-duo of which one is common across language but the other senses may not.
A1 = A1 A2 = Null	Lexical under specification leading to lexical ambiguity. The word has a meaning in one language similar to the other. In addition to that the same word has a specialized sense in the prior which is absent in the later.
A ≠ A ↙ B ≠ B	Semi cross lexical ambiguity is an extension of no overlapping pattern where a pair of words exists in a language-duo and one of the word in the pair connect with the one which are not their replica
A ≠ A ↙↘ B ≠ B	Full Cross lexical ambiguity is an extension of no overlapping pattern where a pair of words exist in a language-duo but both of the words connect with the ones which are not their replicas

Table: 1 Cross Linguistic Patterns

4.2.1 Complete overlapping:

The complete overlap of word senses; we find "Overlapping polysemy" which refers to cases in which items in two languages have exactly the same meanings. In Indian language scenario, normally some words have same origin like proto-Dravidian or Sanskrit borrowed into different languages.

Let us consider an example of overlapping polysemy among Malayalam '*aṭi*' and Tamil '*aṭi*'. The word carries four senses as follows:

1. To Beat (Verb)
2. The part of the leg of a human being below the ankle joint (Noun)
3. The lower part of anything (Noun)
4. A linear unit of length equal to 12 inches or a third of a yard (Quantifier)[1]

We can observe the varying degrees of polysemy exhibited by them and come to the conclusion that the four senses exhibit "Almost complete" overlapping polysemy patterns. Overlapping polysemy poses no problems for multilingual dictionaries.

4.2.2 No Overlapping:

In contrast to the above we observe common phenomena that the word borrowed from the same source into two different languages may have diverging structure. For example '*laːɲʧana*' in Kannada and Malayalam exhibit semantic overlap when it comes to the basic sense 'indication of something, highlighting, marking something'. However they differ widely in their meaning extensions when it comes to more narrowed senses over time. In Kannada '*laːɲʧana*' widely used to describe 'Emblem - A visible symbol representing an abstract idea'[1]. This concept is not carried in Malayalam. But it is carried as 'Indication - Something that serves to indicate or suggest' The Kannada '*laːɲʧana'* cannot be equated with Malayalam '*laːɲʧana*' anymore. No overlapping poses an issue to the lexicographer, so that simply looking into the word and not the sense will not help while connecting words.

4.2.3 Semi Overlapping:

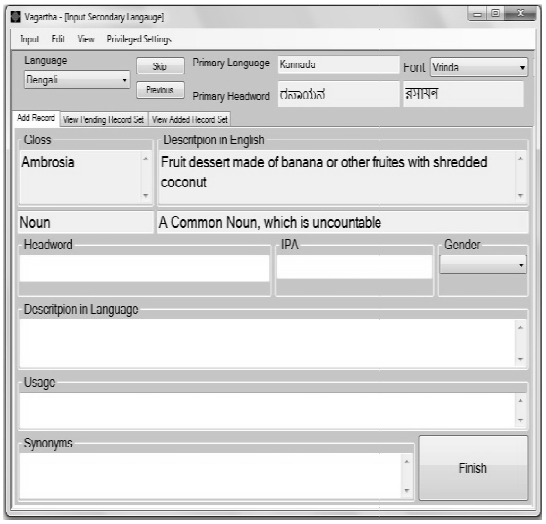
We came across situations in which a word may be having more than one sense in a language-duo of which one is common across language but the other senses may not. For example both Malayalam and Tamil have the word '*kaṯṯi*' and it is used in two senses in both languages. Only one sense is a shared meaning and the other sense is not mutually related. Both words share the meaning 'Knife - A weapon with a handle and blade with a sharp point'[1].

Malayalam '*kaṯṯi*' has a sense 'Burnt - Destroyed or badly damaged by fire' where as Tamil '*kaṯṯi*' has a sense 'Loudly - With relatively high volume'.[1] Simply because these two words are sounds similar and connected in one sense, database architecture should not allow the sense of 'Burnt' to get linked with 'Loudly'. Concept-set-model will take care of it since the words are connected to concepts rather than words. But lexicographer has to be cautious not to jump into conclusions by just looking at the transliterated word that is offered to assist.

4.2.4 Lexical under specification leading to lexical ambiguity:

The fourth type of cross-linguistic phenomenon posing problem for the lexicographer is, cases in which there are no clear equivalents in the target languages. The word has a meaning in one language similar to the other. In addition to that the same word has a specialized sense in the prior which is absent in the later, these cases may lead to zero translations. When the word is outside the culture of the target language and has to be linked, usually lexicographer chooses to borrow the word from source by transliterating the word, (like English word 'tulip' is '*tulip*' in Kannada, as it describes a particular flower.) But in this case lexicographer cannot borrow the word as foreign word for the sake of dictionary entry since it leads to polysemy.

Let's have a look on such a case



Interface created for linking words to concept

'*rasa:jana*' of Kannada and '*rasa:jana*' of Bangla exhibit semantic overlap when it comes to the basic senses describing mixture of two or more elements. It is mainly used for the sense 'Chemical - Material produced by or used in a reaction involving changes in atoms or molecules'[1] in both languages. However they differ widely in their meaning extensions when it comes to more specialized sense. The Kannada '*rasa:jana*' is used to describe 'Ambrosia- Fruit dessert made of bananas and other fruits with shredded coconut'[1]. This concept is not carried in Bangla. To give equivalent, the Lexicographer cannot borrow it easily since it leads to creating confusion because it is not familiar with the language culture. In such cases lexicographer can just describe the concept in Bangla to convey the meaning to the user. Creating or borrowing a word leads to other complications like social acceptance of something which is not at all part of culture.

In spite of its complexity to find proper equivalents for difficult lexical items across, linguistically it is necessary to account for them within the Database. Without their inclusion, neither humans nor machine will be able to successfully use the database for translation purposes.

4.2.5 Semi Co-lexical pattern

Even though a concept is not a lexical ambiguity we observed a potentially confusing pattern for a lexicographer. This is an extension of no overlapping pattern where a pair of words exist in a language-duo and one of the word in the pair connect with the one which are not their replica

For example '*upanja:sa*'and '*ka:ḍambari*' are part of vocabulary of Kannada and Hindi. Both words have Sanskrit origin.

Kannada '*upanja:sa*' is 'Lecture - A speech that is open to the public'[1].

Hindi '*upanja:sa*' is, 'Novel - an extended fictional work in prose; usually in the form of a story'[1].

In Kannada '*ka:ḍambari*' is 'Novel' and in Hindi '*ka:ḍambari*' is 'Cluster-of-Clouds'[2]. Both words are present in both languages. But one of the words is having the meaning of the other but the other words are nowhere associated. Lexicographer should not take these words lightly and connect as per their understanding of the word in their language.

upanja:sa Kannada	≠	*upanja:sa* Hindi	=	*Kadamabari* Kannada	≠	*Kadamabari* Hindi

The Lexicographer has to take care of the context which appears with the word before connecting it into a sense in their language. Mere identifying the word in their own language will not help them anyway.

4.2.6 Full Co-lexical pattern

This is an extension of no overlapping pattern where a pair of words exists in a language-duo, having same origin but both of the words connect with the ones which are not their replicas. It is also a potentially confusing pattern for a lexicographer.

For example, the words '*samɛo:ḍʰana*' and '*anusanḍʰa:na*' is present in both Kannada and Hindi. Both words are having Sanskrit origin. Kannada '*samɛo:ḍʰana*' carries the sense 'Research - Systematic investigation to establish facts'[1]. In Hindi '*anusanḍʰa:na*' is the word for the same sense.

One of the senses that Kannada '*anusanḍʰa:na*' carries is 'Modification- The act of making something different in order to achieve desired format'[1]. And in Hindi, word '*samɛo:ḍʰana*' goes with the sense. The word has other senses like 'examine', 'union' etc in Kannada.

In this case since both words are part of both the languages vocabulary so the lexicographer has to take extra care to look into the context while connecting. Simply looking into the transliteration form offered by the interface to facilitate the lexicographer will not help and may cause wrong connections.

Conclusion:

As per our observations every word is a new word for the lexicographer. A lexicographer has to take appropriate measures not to get mistaken by looking at the source language word. We mentioned our efforts to ensure appropriate management of the multilingual and multidirectional dictionary project. Once developed, such a dictionary provides a vital resource for cross lingual lexicographers and programmers. At present the data building with the approach of concept set modeling is being carried out. Once the substantial data is entered many more complexities and linking issues may be created. Probable solutions for the same are to be researched accordingly.

Reference:

Christophe Roche, Marie Calberg-Challot, Luc Damas, Philippe Rouard. 2009. Ontoterminology: A new paradigm for terminology. International Conference on Knowledge Engineering and Ontology Development. Madeira. Portugal. 321-326.

Hans Christian Boas (Ed.). 2009. Multilingual FrameNets in Computational Lexicography: Methods and Applications. Walter De Gruyter GmbH & Co. Berlin.

Henri Béjoint. 2000. Modern Lexicography. Oxford University Press. Oxford.

Ivan A Derhanski. 2009. Bi-and Multilingual Electronic Dictionaries: Their Design and Application to Low- and Middle-Density Languages. Language engineering for lesser-studied languages. IOS Press.

Leacock C. and Ravin. 2000. Polysemy. Oxford University Press. Oxford.

Susan J. Behrens, Judith A. Parker (Ed). 2010. Language in the Real World. Routledge. Oxon. 67-88.

Rajesha N, Ramya M and Samar Sinha. 2011 Lexipedia: A Multilingual Digital Linguistic Database. Language in India Special Volume: Problems of Parsing in Indian Languages. 52-55. ISSN 1930-2940.

Timothy Baldwin, Jonathan Pool and Susan M. Colowick. 2010. PanLex and LEXTRACT: Translating all Words of all Languages of the World. Coling: Demonstration Volume. 37-40. Beijing.

[1] Wordweb English Dictionary

[2] Lokbharati Brihat Pramanik Hindi Kosh

Tutorial for Deaf – Teaching Punjabi Alphabet using Synthetic Animations

Lalit Goyal, Assistant Professor,
DAV College, Jalandhar.
goyal.aqua@gmail.com

Dr. Viahal Goyal, Associate Professor,
Punjabi University, Patiala.
vishal.pup@gmail.com

Abstract

Developing an automatic tool for educating students has become essential in today's world of computerization. For differently abled people, especially in India, where the resources are scares for educating them, it becomes essential to develop such technologies which give the opportunity to each and every individual to get the education online and free of cost. Research work has been done to create HamNoSys notation of atleast two words corresponding to each alphabet of Punjabi Language. The HamNoSys notation when used in JASigning animation tool produces the synthetic animation. The synthetic animations are better as compared to human videos in terms of memory requirement, standardization, and flexibility. Synthetic animations can be modified as per the requirement whereas the human videos cannot be modified. The only drawback that seems is, these synthetic animations may lack the natural non-manual component of sign. The research work has been incorporated to produce the web portal that displays the Punjabi alphabet along with the picture related to that alphabet and the synthetic animation with which that word is signed in Indian Sign Language. The work is the first of its kind for Indian Sign Language.

Keywords Indian Sign Language, HamNoSys, SiGML, Punjabi Alphabet, Synthetic Animation

1 Introduction

Sign language is the visual spatial language which is used by the differently abled hearing impaired people to communicate. Sign Language is the three dimensional language which uses the 3D space around the signer's body using its hands, arms, body postures, face expressions, and head movements.

Among approximately 7105 known living languages worldwide, there are 141 Sign Languages depending upon the region in the world. There are nearly 72 million people who are hearing impaired among nearly 7 billion people on earth. The situation is worst as 90% of these differently abled people have very limited or no access to education and other information. [1][2].

In India, situation is worse; there are 5072914 persons who suffer from hearing disability. Among them, only 546826 hearing impaired persons are literate below primary education [3] which accounts for only 10.78%.

Sign language is different for different countries/regions as this language is not created but it is evolved by deaf people. So, depending on the region, the signs are different as well as the grammar is also different depending on the region. As far as Indian Sign Language is concerned, it is categorized in manual and non-manual signs which may be static signs or dynamic signs. Static signs are those signs which do not have any movement in their signs. The dynamic signs are those which use the movement of the hands, and the non-manual features of the sign. Most of the signs used in the Sign Language are dynamic signs.

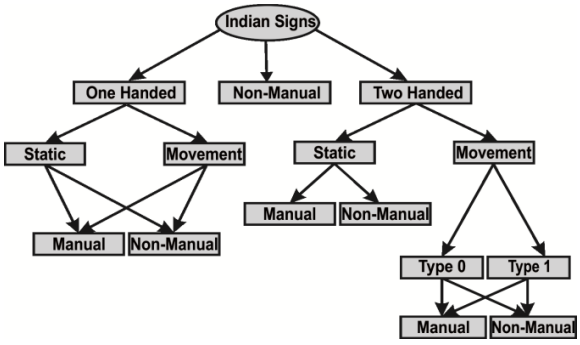


Figure 1: ISL Type Hierarchy

The one handed signs are represented by a single dominating hand whereas the two handed signs are represented by both the hands of the signer.

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 172–177

Both one handed as well as two handed signs can be either static or dynamic (having movements). Each of the static and movement signs is further classified into manual and non-manual signs.

Two handed signs with movements can be further distinguished as: Type0 and Type1 signs.**Type0** signs are those signs in which both the hands are active i.e. both the hands are in motion.**Type1** signs are those signs in which one hand (dominant) is more active as compared to the other hand (non-dominant).

2 Review of Literature

The education is essential for the good social life. The hearing impaired persons also required to be educated so that they can communicate with each other in their own language as well as capable of communicating in their social life. In India, various schools are available to impart education to these people but these schools are limited to only urban areas. Even the number of schools is scarce in urban area. A lot of hearing impaired persons have no access of education and so they are cut off from the society.

A lot of work has been done in implementation of sign language dictionary worldwide. Dictionaries have been created in the form of books which are obsolete in this day and age of computerization. Video dictionaries are available for sign languages of many countries like America, Britain, Australia, Spain, Italy, and even India. These video dictionaries can be categorized as real character (human being) producing the sign or computer generated animated character (avatar) producing the sign. A lot of Indian sign language dictionaries are available which uses the real human being producing the sign for an English word. No Indian sign language dictionary is available which uses computer generated character (avatar) technology.

- The Ramakrishna Mission collaborated with CBM International, Germany for a project on sign language dictionary in year January 1999. The goal for the project was to standardize Indian Sign Language. On November 24, 2001, the first Indian Sign Language dictionary was released which contains more than 2500 signs. These signs were taken from 42 cities in 12 States to provide a common sign language all over India[4]. The signs in this online dictionary are videos of real human English Alphabets are also included in the sign dictionary. This dictionary does not contains any other languages of India even Hindi alphabets are not included.
- Spread the Sign, an international project by Leonardo da Vinci supported by the European Commission through the Swedish International Program Office of Education and Training. The goal of this project is to share various sign languages from different countries over the internet. The drawback of this work is that it has videos for the words rather than animations which take a long time to load as compared to synthetic animations [5]. The participation of various countries is not upto the mark.
- Handspeak created is the American Sign Language dictionary. The dictionary is released on the domain handspeak.com in 2000. The website contains the ASL signs, some variants of ASL signs, some verb inflections, and more. The dictionary is produced and signed by native ASL bilinguals [6].
- Sign Smith [7] is a 3D illustrated dictionary of ASL. It is used as educational software for the hearing impaired people of America. It is also an authoring tool to create ASL content.

A lot of work is done in developing the dictionaries of sign language of various countries but no work has been found in developing the dictionary or tutorial for teaching the Punjabi (Language of the state Punjab) alphabet.

3 Punjabi Alphabet

Punjabi is one of the 22 official languages of India which is spoken in the state of Punjab. In India, Punjabi is written in Gurmukhi alphabet which is composed of vowels, consonants, vowel diacritics.

The vowels and vowel diacritics are known as Laga Matra where as consonants are known as Vianjans. In Gurmukhi alphabet, there are a total of 10 vowels, 10 vowel diacritics and 41 consonants.

Vowels

ੳ ੳੂ ੳ ਅ ਆ ਐ ਔ ਏ ਇ ਈ

Vowel Diacritics

ਾ ਾ ਂ ਿ ੀ ੇ ੈ ੋ ੌ ੁ ੂ

Consonants

ੳ ਅ ੲ ਸ ਹ
ਕ ਖ ਗ ਘ ਙ
ਚ ਛ ਜ ਝ ਞ
ਟ ਠ ਡ ਢ ਣ
ਤ ਥ ਦ ਧ ਨ
ਪ ਫ ਬ ਭ ਮ
ਯ ਰ ਲ ਵ ੜ
ਸ਼ ਖ਼ ਗ਼ ਜ਼ ਫ਼ ਲ਼

Other Symbols

ੰ ੱ ੵ ੍ ੁ ੂ

4 Real Vs Synthetic Video Dictionaries

Translation process from a source language to target language requires a bilingual dictionary between the source and target languages. In case of translating Punjabi text to Indian Sign Language, bilingual dictionary of Punjabi word and Indian Sign Language is required. Punjabi-ISL bilingual dictionary is completely different from any other bilingual dictionary between the spoken languages. The reason behind is that the Indian Sign Language is the visual spatial language which cannot be spoken or written. So, irrespective of bilingual dictionaries of other spoken languages, for each Punjabi word, the corresponding ISL word is not the written word. Here, the Punjabi word's counterpart in ISL can be a real human video, sign picture, coded sign language text, or synthetic animation. All the approaches have their own pros and cons but the synthetic animations are well suited because of scalability of computer generated avatar. A comparison of all the media used for creating the bilingual dictionary of English-SL has been given in the following table:

Table 1: Comparison of Different Media for Representing the Sign

Kind of Media	Pros	Cons
Video Signs	• Realistic • Easy to create	• Time consuming to create • High memory consumption • Not supported by translation system
Pictures	• Very less memory consumption	• Time consuming to create pictures • Not realistic as compared to videos • Not supported by translation system
Coded Sign Language Text (written form of SL)	• Minimal Memory consumption • Supported by translation system as it is the written form and can be processed very easily	• Very difficult to read and understand • Required to be learnt
Synthetic Animations	• Very less memory consumption • Can be easily reproduced • Supported by translation system • Avatar can be made different according to choice	• Not as realistic as human videos.

5 HamNoSys Notation

Sign Language is a three dimensional language which cannot be written just like the other spoken languages like English, Hindi, Punjabi etc. But, researchers have created various writing notations of sign language. The benefit of writing notation is that the translation process from a spoken language to sign language becomes feasible. Even a writing notation of sign language is must for creating the computer

generated character (Avtar) that can be animated just like the human being. Various writing notation available for writing the sign language are Bébian Notation, Stokoe Notation, Gloss Notation, Hamburg Notation System (HamNoSys), SignWriting (SW), si5s, SignFont, SignScript, SLIPA etc. We have used HamNoSys notation to create the animation of the words related to Punjabi alphabet.

The Hamburg Notation System (HamNoSys) is a phonetically based notation system that was developed by Siegmund Prillwitz in 1984[8] at the institute of German Sign Language, University of Hamburg. HamNoSys notation is rooted in the Stokoe notation with more detail handling the non-manual component of the sign also. Around 200 symbols are available in this notation system to describe any sign. The structure of this notation contains mainly four components: Symmetry operator (in case both the hands are used), NMF (to describe the non-manual features), Initial Configuration (contains in sequence the hand shape, hand orientation, and hand location), and Action/Movement (the dynamic part or movement of the hands)

The syntax of HamNoSys notation is the sequence of symbols of symmetry, non-manual features, hand features (hand shape, orientation, location) and last the hand movements. Following figure shows the basic structure of the HamNoSys notation. The first component of HamNoSys notation is always the symmetric operator which is used for two handed signs. The second component is for non-manual part of the sign such as face expressions, head movement, body movement, lips movement (for getting the phonetic expression). The third component is for hand shape, hand orientation, and hand location. The forth component of the notation is the movement of the hands in case of dynamic signs.

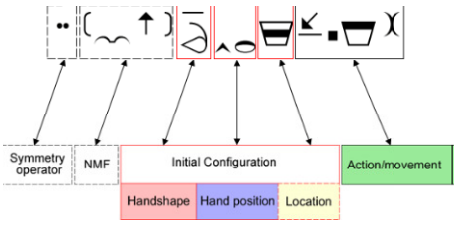


Figure 2: Structure of the HamNoSys

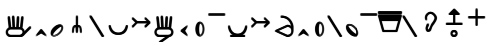


Figure 3: HamNoSys Notation for word "Beautiful"

An advantage of HamNoSys is that it is international and can be used to write any Sign Language. This notation system was initially handwritten, but a machine readable Unicode is now available from the University of Hamburg. This notation is iconic, has a formal syntax as shown above and can be stored in a computer database. The limiting part of this notation is that it does not provide an easy way to describe non-manual features, such as facial expressions and body movements but still the non-manual part produced by using this notation is comparatively better than other notations.

An XML encoding of HamNoSys called Signing Gesture Markup Language (SiGML) is also available. SiGML encoding is used to produce the animation of the sign using an animation tool JASigning [11]. It was developed for the ViSiCast project by Richard Kennaway [9]. Some of the symbols used in HamNoSys notation are:

Figure 4: Symbol Set used in HamNoSys Notation System

6 Tutorial for Punjabi Alphabet

Teaching Punjabi alphabet to hearing impaired students is very hard because of lack of teaching resources. We have tried an attempt to produce the web portal which displays the Punjabi alphabet along with a word for that alphabet. Along with the textual information (Punjabi

Alphabet and corresponding word), a picture of the word is also displayed. The animation is produced in ISL describing how to produce the sign for each word. For each Punjabi alphabet, we have chosen two word for better understandability. For a total of 31 alphabets, we have created HamNoSys code for 61 words. HamNoSys has an alphabet of about 200 symbols (Unicode of this notation system is available) which covers almost all the hand shapes, hand location, hand/palm orientation, hand movement, and non-manual parts of the sign. Later this HamNoSys can be converted into SiGML (Signing Gesture Markup Language) tags which are sort of XML tags that can be animated by an animation tool using an Avatar. The sequence of steps for creating the animation from English word is as shown in the following architecture [10]:

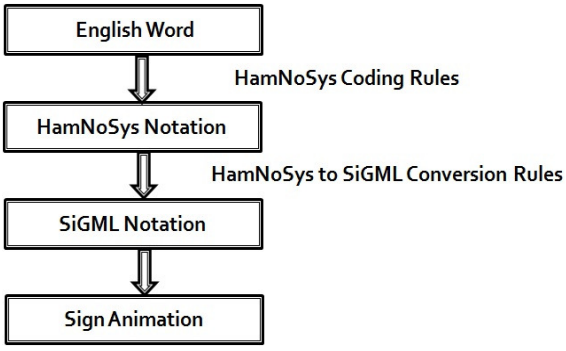


Figure 5: Architecture to Produce the Animation from English Word

All the 61 signs are dynamic signs except the sign of word **ਨੱਕ** (nose) which is static single handed sign. Below table shows the list of words corresponding to each Punjabi alphabet.

Table 2: Words with each Alphabet coded in HamNoSys

S.No.	Punjabi Alphabet	Words
1	ੳ	ਉਠ, ਉਂਗਲੀ
2	ਅ	ਅੱਖ, ਅੰਬ
3	ੲ	ਇੱਲ, ਇੰਜਣ
4	ਸ	ਸੇਬ, ਸੰਤਰਾ
5	ਹ	ਹਾਥੀ, ਹਿਰਨ
6	ਕ	ਕੇਕ, ਕੁੱਤਾ
7	ਖ	ਖਿੜਕੀ, ਖੰਬ
8	ਗ	ਗਾਂ, ਗੰਡਾ
9	ਘ	ਘਰ, ਘੋੜਾ
10	ਚ	ਚੰਨ, ਚਿੜੀ
11	ਛ	ਛੱਤਰੀ, ਛੱਲਾ
12	ਜ	ਜਹਾਜ, ਜੰਗਲ
13	ਝ	ਝੰਡਾ, ਝਰਨਾ
14	ਟ	ਟਾਰਚ, ਟੋਕਰੀ
15	ਠ	ਠੇਲਾ, ਠੰਡ
16	ਡ	ਡੱਡੂ, ਡੱਬਾ
17	ਢ	ਢੋਲ, ਢਾਲ
18	ਤ	ਤਿੱਤਲੀ, ਤਰਬੂਜ
19	ਥ	ਥਾਲੀ, ਥੈਲਾ
20	ਦ	ਦੰਦ, ਦਰਖ਼ਤ
21	ਧ	ਧਰਤੀ, ਧਾਗਾ
22	ਨ	ਨੱਕ, ਨਾਰੀਅਲ
23	ਪ	ਪਤੰਗ, ਪਾਣੀ
24	ਫ	ਫੁੱਲ, ਫਲ
25	ਬ	ਬੈਲਗੱਡੀ, ਬਤਖ
26	ਭ	ਭੇੜੀਆ, ਭਾਲੂ
27	ਮ	ਮੱਖਣ, ਮਛਲੀ
28	ਯ	ਯੋਗ
29	ਰ	ਰੇਲ, ਰਾਤ
30	ਲ	ਲੋਮੜੀ, ਲੜਕੀ
31	ਵ	ਵਾਲ, ਵਰਖਾ

Figure 6 : Screenshot of the Punjabi Alphabet Tutorial

7. Conclusion

Automatic tool for learning Punjabi Alphabet by the hearing impaired people is challenging task because creation of synthetic animation for all the words corresponding to Punjabi alphabets is very difficult to create. This paper represents the creation of synthetic animations using HamNoSys notation for 61 words for all the Punjabi alphabet. All the synthetic animations are incorporated in the web portal. The present work is important for hearing impaired people

because of scarce resources like deaf schools in India. The tool can be very much beneficial for imparting education to these differently abled people.

References

[1] Ethnologue: Languages of the World. (2015). Retrieved July 10, 2016, from http://www.ethnologue.com/

[2] WFD | World Federation of the Deaf - World Federation of the Deaf, WFD, human rights, deaf, deaf people. (2015). Retrieved July 10, 2016, from https://wfdeaf.org/

[3] Disabled Population by type of Disability, Age and Sex - C20 Table. (2011.). Retrieved March 21, 2016, from http://www.censusindia.gov.in/2011census/C-series/c-20.html

[4] FDMSE-Indian SIGN LANGUAGE. Available from: http:// enabled.in/wp/indian-sign-language-dictionary-website/

[5] European sign language center. Available from: http://efsli. org/

[6] Handspeak. Available from: https://prezi.com/fugyte-fvya6z/httpwwwhandspeakcomwordwwhomp4/

[7] VCOM3D. Sign smith products. Available from: http:// www.vcom3d.com

[8] Hanke, T. (2004, May). HamNoSys-representing sign language data in language resources and language processing contexts. In *LREC* (Vol. 4).

[9] Kennaway, R. (2001, April). Synthetic animation of deaf signing gestures. In *International Gesture Workshop* (pp. 146-157). Springer Berlin Heidelberg.

[10] Goyal, L., & Goyal, V. (2016). Development of Indian Sign Language Dictionary using Synthetic Animations. In *Indian Journal of Science and Technology*, 9(32).

[11] JASigning. (2015). Retrieved October 15, 2016, from http://vh.cmp.uea.ac.uk/index.php/JASigning

SemTagger: A Novel Approach for Semantic Similarity Based Hashtag Recommendation on Twitter

Kuntal Dey[†], **Ritvik Shrivastava**[⋆], **Saroj Kaushik**[§], **L Venkata Subramaniam**[†]
[†]IBM Research, India; [⋆]NSIT Delhi, India; [§]IIT Delhi, India
[†]{kuntadey, lvsubram}@in.ibm.com; [⋆]ritviks.it@nsit.net.in; [§]saroj@cse.iitd.ac.in

Abstract

This paper proposes a semantic similarity based novel approach, to assign or recommend a hashtag to a given tweet. The work uses a Latent Dirichlet Allocation (LDA) based learning approach. In the training phase, we learn the latent concept space of a given set of training tweets, via topic modeling, and identify a group of tweets that act as representatives of the topic. In the inference phase, we create a probability distribution of a given test tweet belonging to the learned topics, and find the semantic similarity of the test tweet with representative tweets for each topic. We propose two assignment approaches. In one approach, we assign hashtags to a target tweet, by obtaining these from a set of representative training tweets, that have the highest semantic similarities with the target tweet. In the other approach, we combine (a) the semantic similarity of the target tweet with the representative tweets, and (b) the assignment probability of the target tweet to a given topic, and assign hashtags using this joint maximization. The hashtags are assigned to the target tweet, by selecting the top-K values from the combination. Our system yields F-score of 46.59%, improving over the LDA baseline by around 6 times.

1 Introduction

1.1 Background and Motivation

The hashtag recommendation problem for Twitter addresses suggesting appropriate hashtags to a user for assigning to a tweet they would post. Recommendation of hashtags for Twitter messages has emerged as a mainstream area of research. Practically, only around 10-15% Twitter data tends to have hashtags, as observed by (Hong et al., 2011). And yet, as observed in the literature, hashtags play a critical role in solving significant problems, e.g., information diffusion (Starbird and Palen, 2012) (Tsur and Rappoport, 2012), topic modeling (Asur et al., 2011) and many other problems as observed by the literature survey conducted by Dey et al. (2017). All of the above indicate that it is important to solve the problem of hashtag recommendation.

The problem has been received with strong research enthusiasm in recent times. Several research solutions have been proposed. Some early-breaking works include the works by Zangerle et al. (2011), Ding et al. (2012) and Ding et al. (2013), that follow approaches such as *tf-idf* and translational models. Several other approaches emerged over time. Topical models, such as Zhang et al. (2014) and Gong et al. (2015), started finding way into the literature. Deeper and more focused methods started getting proposed, such as recommending hashtags for tweets containing a hyperlink by (Sedhai and Sun, 2014). Subsequently, deep neural network based models emerged. Weston et al. (2014) predicted hashtags using a convolutional neural network (CNN) (Krizhevsky et al., 2012) based approach, and learned semantic embeddings of hashtags. Gong and Zhang (2016) used CNN with attention mapping. They attained an F-score of 39.8%, which is the best in the literature till date.

1.2 Central Idea

We observe that, while Dirichlet and specifically Latent Dirichlet Allocation (LDA) (Blei et al., 2003) based approaches exist in the literature to solve the problem at hand, these works tend to model the topics appearing in a given target tweet as a semantic (topical) alignment with the training tweets, and use the hashtags appearing in those tweets for recommendation. An important aspect

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 178–187

that appears unexplored is the semantic similarity of the target tweet, with the training tweets that are topically aligned. In the current work, we hypothesize that, considering the semantic similarity of the training tweets that are topically (LDA-wise) based aligned to the target tweet, and assigning hashtags to the target tweet using this similarity, is an effective methodology for recommending hashtags to tweets.

In the training phase, we use a LDA-based topic modeling, to learn the semantic concept space covered by the training tweets, and identify topics via topic modeling. We identify a group of tweets that act as representatives of the topic. For inference (assigning hashtags to a given target tweet), we create a probability distribution of the target tweet belonging to the learned topics. We subsequently find the semantic similarity of the target tweet with representative tweets for each topic, using a state-of-the-art model externally learned specifically for Twitter (Dey et al., 2016).

We propose two variants for making the recommendation. In one variant, we recommend hashtags to the target tweet, using the semantic similarity of the target tweet with the representative tweets for each topic derived, and picking from the more similar training tweets. In the other variant, we combine (a) the semantic similarity of a target tweet with the representative tweets for each topic derived, and, (b) the assignment probability of the target tweet to a given topic, to obtain a combined score of each representative tweet (across the different topics) to get selected. We rank the representative tweets based on the score of combination, and recommend hashtags based upon the hashtags observed in the top-K ranked tweets. We empirically determine K as 3, and observe that our methodology produces highly effective results, lifting the F-score by around 6 times from the LDA baseline.

1.3 Our Contributions

The contributions of our work are the following.

- We provide a novel methodology to address the problem of hashtag recommendation on Twitter. Our approach replies upon recommending hashtags to a given target tweet, based on semantic similarity of the target tweet with topically similar training tweets.
- We propose *SemTagger*, a framework where we learn the latent concept space of a given set of training tweets, via topic modeling, and assign hashtags to test tweets using (a) a combination of the semantic similarity of a test tweet with representative training tweets, and the assignment probability of the test tweet to a given topic, and (b) assigning hashtags by selecting the top-K values from the combination thus computed.
- We empirically determine the effectiveness of the proposed approach. In our experiments, we observe that our methodology delivers an F-score of 46.59%, which is around 6 times higher compared to a corresponding LDA baseline of 7.79%.

The rest of the paper is as follows. Section 2 provides an overview of the literature in the space of Twitter hashtag recommendation. This is followed by the details of our methodology in Section 3. Section 4 presents the experiment design and results. Section 5 is used for a brief discussion of a few aspects of interest. The paper is finally concluded in Section 6.

2 Related Work

Hashtag recommendation has been established as a well-accepted research problem for nearly a decade now. Multiple approaches have been proposed by researchers exploring the problem from several aspects. In an early work, while solving a sentiment classification problem, Davidov et al. (2010) had attempted to address hashtags indicative of sentiments. However, the first-ever work that focused completely on hashtag recommendation, was carried out a year later, by Zangerle et al. (2011). In this work, the authors used the *tf-idf* approach to compare tweet-pair similarity, and thus computed the similarity of a target tweet with given training tweets. They subsequently retrieved tweets with the most similar messages, and heuristically ranked and recommended the hashtags that appeared in the extracted tweets. In a body of works that followed, Ding et al. (2012) and Ding et al. (2013) converted the hashtag recommendation to a translation problem. Their model is centered around an unsupervised learning method using a latent variable estimation based topical translation model. They hypothesize that hashtags and trigger words of tweets are two different languages with the same meaning that occur in parallel. They use "topic-specific word trigger to bridge the vocabu-

lary gap between the words in tweets and hashtags, and discovers the topics of tweets by a topic model designed for microblogs".

Subsequently, a large number of research works started emerging in the literature, that attempted to solve the problem. Several novel approaches were proposed, covering different aspects of the problem. One such work, that attempted to recommend hashtags only to the tweets containing a hyperlink in the content, was proposed by Sedhai and Sun (2014). Their approach consisted of two phases. In the first phase, they selected a set of candidate hashtags using the attributes computed from tweet content, such as hyperlinked documents, named entities contained in the referred webpage as well as present in the tweet, and the domain of the content of the webpage that the hyperlink refers to. In the second phase, they formulate as a learning-to-rank problem, and solve with RankSVM to aggregate and rank the candidate hashtags selected in the first phase.

Gong et al. (2015) proposed a Dirichlet based method. They adopted a Dirichlet based mixture model, incorporating types of hashtags as hidden variables. Motivated by Liu et al. (2012) and philosophically akin to Ding et al. (2012) and Ding et al. (2013), they also model assuming that hashtags and tweet content are parallel descriptions of the same content.

A topic-based hashtag recommendation method was proposed by She and Chen (2014). This work treated hashtags as topic labels, and performed supervised topic model learning over these labels, to discover inter-word relationships. They treated the words as one of two types: background words that are prevalent in many of the tweets, and local topic words that are more specific to that topic. They inferred the probability that a hashtag will be contained in a new tweet, and generated hashtags for recommendation using a symmetric Dirichlet distribution of the local and background words. Zhang et al. (2014) proposed another topic-based hashtag recommendation method. Their work used a topical model based method, incorporating both temporal and personal information. They extended over the well-established translational model for hashtag recommendation. They divided the time horizon into T epochs, and analyzed at a per-epoch level to ensure temporal relevance of recommended hashtags. They drew from a multinomial word-topic distribution and recommended the hashtags that have the maximum probabilities in the draw. Among other works, Godin et al. (2013) too proposed another effective topic-based hashtag recommendation method.

The recent advances in deep neural network based learning (deep learning), has motivated researchers to attempt such techniques on the hashtag recommendation problem. In an early application of deep learning on this problem, Weston et al. (2014) predicted hashtags using a convolutional neural network (CNN), and learned semantic embeddings with hashtags. They posed as a supervised learning problem, treating the hashtags as labels assigned to the tweet content. Their model represents the words, as well as the entire textual posts, as embeddings in the intermediate layers of their deep-CNN architecture. The recent work by Gong and Zhang (2016) used CNN with attention mapping. They, too, converted the words into embeddings, and used a local small window based attention map, where each given window surrounds a word around which the attention is provided. They attained an F-score of 39.8%, lifting the performance over a LDA baseline by 6.42 times, making the work the most effective hashtag recommendation system known in the literature.

Our work uses the LDA-based models, but introduces a novel mechanism of augmenting topical similarity with semantic similarity of target and training (known) tweets. This approach is the first of its kind, and it outperforms the systems known in the literature except the work by Gong and Zhang (2016). However, the practicability of deep learning in real-life systems that are often used from mobile phones, remains a question till date. Deep learning on mobile phones has remained a challenge [1] that has not been addressed till date in a satisfactory manner. And yet, 85% of the total usage time on Twitter happens on mobile phones[2]. Our approach is lightweight, making it practical and useful in real life, including being usable from mobile phones. Thus, while in terms of performance (F-score) metrics our model is second to a deep-learning based model (Gong and Zhang, 2016), practically, not counting the deep learning systems that are not fit for use in real-life solutions that often are executed on mobile phones, our work establishes a new real-life benchmark.

[1] https://conferences.oreilly.com/strata/strata-ca-2017/public/schedule/detail/56179

[2] https://twitter.com/wsjtech/status/451886622788055040?lang=en

3 Details of Our Approach

We use a topic modeling and semantic similarity driven approach to model our solution framework. The details of *SemTagger*, our framework, are presented below.

3.1 Data Cleaning

The very first step followed in the training as well as inference phases, is data cleaning. This comprises of the following steps.

- **Removal of tweets without any hashtag:** In order to train our model, we need tweets that necessarily contain hashtags. Further, since the objective of the present work is to perform hashtag recommendation, the target (test) tweets that we shall assign hashtags to, will also need to contain ground-truth hashtags assigned by the user posting the tweet. The testing will be performed by hiding the hashtags from the target tweets and assigning the predicted hashtags to these tweets using our model; however, the performance of our model will be validated by the ground-truth hashtags that were hidden. Thus, all the tweets we use for our process necessarily need to contain at least one hashtag. Driven by this requirement, we retain only those tweets that contain at least one hashtag, and eliminate the remaining tweets.

- **Non-English tweet removal:** Since the focus of our work is around tweets authored in the English language, we eliminate the non-English tweets from our dataset. The language-marker field present in the raw Twitter data indicates the language of each given tweet, which is used to detect whether a given tweet is in English or not. This frees our dataset from extraneous and non-useful tweets, and retains only the English tweets that are of interest.

- **Non-ASCII character removal:** Since the non-ASCII characters do not add value to the work, we eliminate the non-ASCII content present in each given tweet (that has been retained otherwise), and retain the remaining part of the text.

After the data cleaning process, we are left with only English tweets, with at least one hashtag, and containing no non-ASCII character.

3.2 Preprocessing

Both in the training and testing phases, we first preprocess the dataset. This includes performing the following operations on each tweet:

1. **Tweet normalization:** We normalize tweet content, by resolving many colloquial on-the-net expressions appearing as part of user-generated social media text, but do not appear in any traditional dictionary. For instance, what appears as *aaf* on Twitter, is expanded to *as a friend* after the tweet normalization process. We normalize the tweets using a net slang dictionary[3] and Han-Baldwin normalization corpus.

2. **Stopword removal:** Stopword removal is an essential step of our process. This step ensures that the superfluous words with practically no information content for the task under consideration are eliminated (such as prepositions, article *etc.*). We perform stopword removal using an online dictionary[4].

The architecture of the data cleaning and preprocessing phases are given in Figure 1.

3.3 Topic Model-Based Training

We perform topic model-based training from the given tweets, to construct a topic distribution model. We subsequently identify a representative set of tweets for each of the topics detected. The training pipeline has been illustrated in Figure 2.

3.3.1 LDA-Based Topic Modeling

We perform LDA-based topic modeling on the training tweet set. This is performed over two steps.

First, a document is created as a concatenation of all the tweets present in the training dataset, minus the hashtags. That is, for a given set of tweets $T = \{t_1, t_2, ..., t_n\}$, containing hashtags $H = \{h_1, h_2, ..., h_m\}$, a document D is constructed as

$$D = \bigcup_{i=1}^{n} t_i - \bigcup_{j=1}^{m} h_j \quad (1)$$

Next, the document is processed for LDA-based topic modeling, and a set of topics $Z =$

[3] http://www.noslang.com/dictionary

[4] https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html

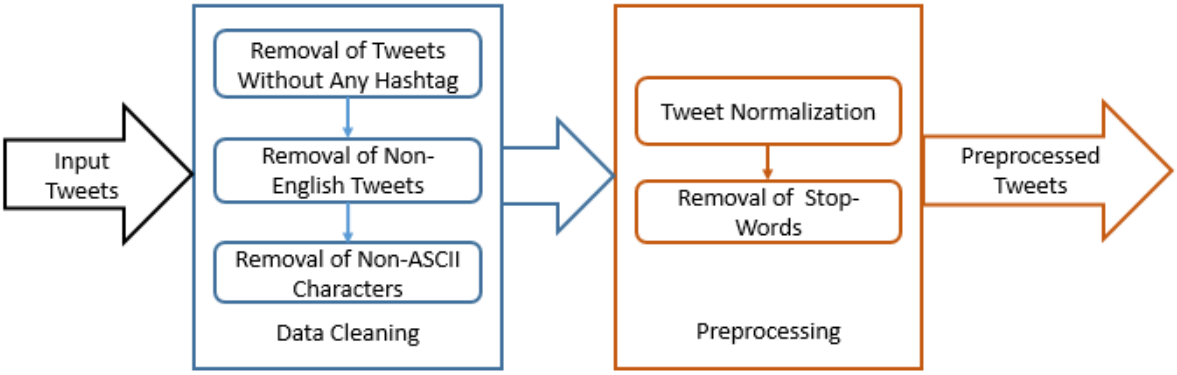


Figure 1: Data Cleaning and Preprocessing

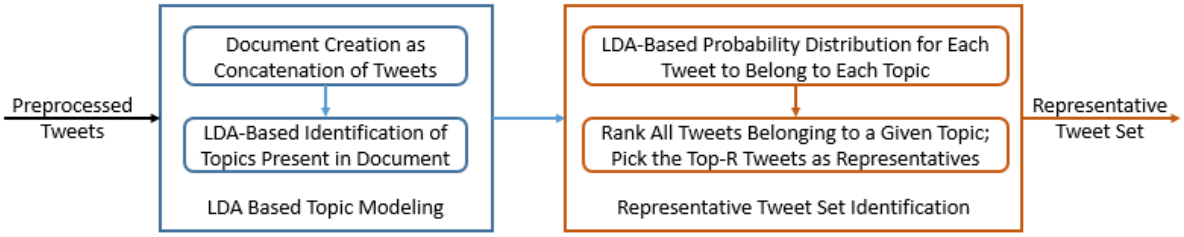


Figure 2: Training and Representative Tweet Set Identification

$\{z_1, z_2, ..., z_l\}$ are learned. Please note that, LDA (Blei et al., 2003) is traditionally modeled as a joint distribution in the following manner:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D} = \prod_{i=1}^{K} p(\beta_i). \prod_{d=1}^{D} p(\theta_d). \prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n}) \quad (2)$$

Here, $\beta_{1:K}$ represent the topics where each β_k is a distribution over the given vocabulary, θ_d are the topic proportions for document d, $\theta_{d,k}$ is the topic proportion for topic k in document d, z_d are the topic assignments for document d, $z_{d,n}$ is the topic assignment for word n in document d, and w_d are the observed words for document d. This process learns the semantic concept space of the training tweets, in form of latent topics.

3.3.2 Representative Tweet-Set Identification

We identify a set of tweets that act as representative tweets for each identified topic. For this, we generate the probability distribution of each tweet to belong to each topic derived, using LDA on the tweet content. For each topic, we rank the tweets by the probability value that a tweet belongs to the topic. We finally pick all the tweets ranked within the top R, to form a representative tweet set of size R for that topic. The output of the training process constitutes of a set of topics $Z = \{z_1, z_2, ..., z_l\}$, a set of representative tweets $T_{z,L} = \forall(l \in L)\{t_{z_l}\} = \forall(l \in L)\{t_{1,l}, t_{2,l}, ..., t_{n,l}\}$ associated with each topic.

3.4 The Hashtag Recommendation Methodology

After topic training and representative tweet set identification, the system becomes capable of assigning hashtags to target (test) tweets provided as input. For this, we first create a probability distribution of a given test tweet belonging to the learned topics. This, again, is performed by generating the LDA-based probability distribution of the tweet content, that quantifies "how much" a tweet belongs to each topic. Using this baseline, we propose a few variants (heuristics) based upon semantic similarity detection, to perform hashtag assignment to each given test tweet. We broadly categorize these approaches in two categories: *semantic similarity based* and *joint probability maximization based* hashtag recommendations.

3.4.1 Semantic Similarity Based

The first method we propose is a semantic similarity rank based hashtag recommendation. Figure 3

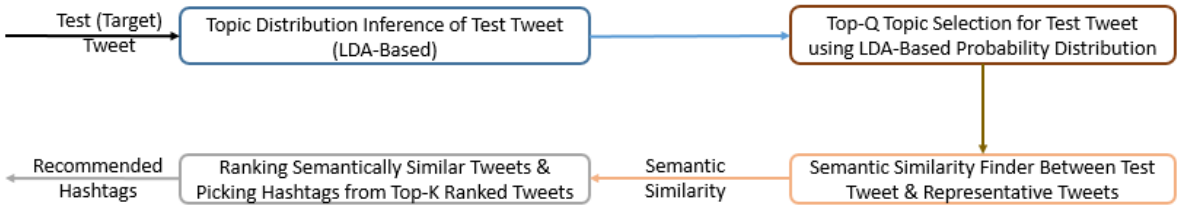


Figure 3: Semantic Similarity Based Hashtag Recommendation

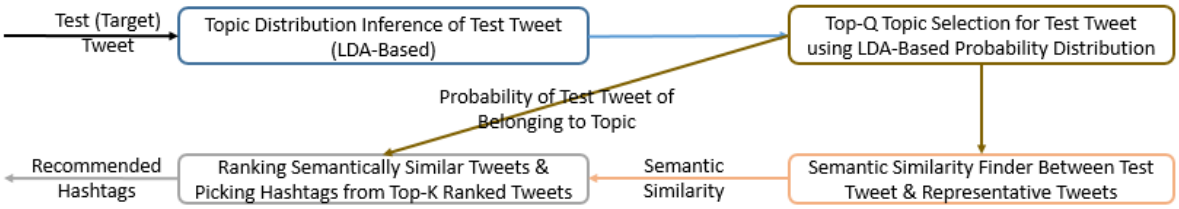


Figure 4: Joint Distribution Maximization Based Hashtag Recommendation

provides a block-level illustration of this method. In this approach, we first select the best (highest probability) Q topics out of Z, using the probability distribution of the given test tweet. We measure the semantic similarity between the test tweet and all the R representative tweets T_{Z_q} across all the top Q topics Z_q. For measuring semantic similarity, we use a transfer learning approach: we use an external semantic similarity learning model given by Dey et al. (2016), which was specifically trained for semantic similarity quantification on Twitter. We rank all the $R*Q$ representative tweets by their semantic similarity scores with the test tweet, and select the hashtags given by the top-K ranked tweets, where K is an externally specified integer.

3.4.2 Joint Distribution Maximization Based

The second method we propose uses the semantic similarity based model as the baseline; however, unlike the earlier approach which was topic distribution-agnostic for ranking the semantically similar tweets, this is topic distribution-aware. Figure 4 provides a block-level illustration of this method. Here, we maximize the combination of (a) the semantic similarity of the test tweet with the representative tweets of a topic, and, (b) the assignment probability of the test tweet to the topic. If the assignment probability of a test (target) tweet t to a topic z_l is $P(t, z_l)$, and the semantic similarity of a test tweet t with one given representative tweet t_j of a topic is $SS(t, t_j)$, then, the combined score for each <test tweet, representative tweet> pair is:

$$CS(t, t_j) = P(t, z_l) \times SS(t, t_j) \quad (3)$$

We rank the $CS(t, t_j)$ values thus obtained, and pick the top-K tweets based upon this rank to select hashtags for the task of recommendation. Thus, in this case, the semantic similarity values of the representative tweets with the test tweet, are not ranked directly; instead, first, the semantic similarity values are combined (multiplied) with the probability of the test tweet belonging to that topic, and then, this combination (product value) is ranked. We assign the hashtags by selecting the top-K tweets in a decreasing (ranked) order of product values, thus inherently selecting the maximal values from the combined distribution.

The overall process that we follow, is given in Algorithm 1.

4 Experiments

We present the details of the experiments conducted and results obtained below.

4.1 Data Description and Tools Used

Using Decahose[5], we collect 10% random sample of all the tweets made on Twitter for 31^{st} January, 2016, and retain all the English tweets that have at least one hashtag associated. We remove the retweets and quoted tweets from both the training and test tweets, as it is trivial to assign hashtags to such tweets, given the actual or recommended hashtags to the corresponding original tweets. We

[5]https://gnip.com/realtime/decahose/

clean the data to remove all hashtags that are simple stopwords [6], and remove the tweets that comprise of only such hashtags (if a tweet has other hashtags too, we retain it). Further, we empirically retain all the tweets that use at least one hashtag which has been used between 200-500 times in the original dataset. This produces a set of $251,649$ English tweets with at least one hashtag. We randomly split into three sets: $175,000$ for training, $25,000$ for validation and the remaining $51,649$ for testing. We evaluate the effectiveness of our system by comparing the recommended hashtags with the actual hashtags present in the test tweets. The dataset details are presented in Table 1.

Tweet Selection Criteria	Count
Total tweets	$34,114,982$
English tweets	$13,410,808$
Tweets with at least one hashtag	$2,417,163$
Hashtag count based retention	$251,649$
Training tweets	$175,000$
Validation tweets	$25,000$
Testing tweets	$51,649$

Table 1: Data description

We use the Stanford NLP Toolkit (Manning et al., 2014) for PoS tagging, Porter stemmer (Porter, 2001) for stemming the tweets, MALLET (McCallum, 2002) for training the LDA based topic models, and Weka (Hall et al., 2009) for running the semantic similarity model.

4.2 Experimental Results

To evaluate the performance of our system, we use precision (Pr), recall (Re) and F-score (F1), computed as $Pr = \frac{N_c}{N_s}$, $Re = \frac{N_c}{N_t}$ and $F1 = \frac{2.Pr.Re}{Pr+Re}$, where N_c and N_s are the correct and total number of hashtags recommended for a given tweet respectively, and N_t is the total number of hashtags present in the semantically similar training tweets under consideration. In an embodiment of our methodology where the number of hashtags to be predicted in the test tweet is provided as an input, we perform experiments by limiting our system to predict the required number of hashtags. We empirically choose the size of the representative tweet set $R = 100$; as well as, we empirically pick the top $Q = 3$ topics that a test tweet is aligned to.

[6] https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html

4.2.1 Selecting K

Experiment	F1(%)	Experiment	F1(%)
Top 1	36.67	Top 2	42.93
Top 3	**46.59**	Top 4	36.52

Table 2: Selecting the value of K using F-scores

Next, we select K, the number of representative tweets to consider for computing semantic similarity with the test tweet. In order to select an effective value of K, we vary the value of K from 1 to higher values, and observe the impact of the values on the final F-score that our system produces. Specifically, we use the semantic similarity match based methodology described in Section 3.4.1. As clear from Table 2, the impact of considering a larger number of semantically similar representative tweet for comparison with a test tweet, is the most effective for $K = 3$. Hence, we choose the value of $K = 3$ for the subsequent experiments.

4.2.2 Joint Distribution Maximization

We compute the combination of the semantic similarity of a test tweet with the given representative tweets of the topics, and the probability of the representative tweets to belong to the respective topics, using Equation 3. The scores are ranked, and we pick the tweets that are ranked in the top-K to select hashtags for the task of recommendation. We empirically observe K=3 to deliver the best performance, wherein, the F1-score is **46.28%**, precision 34.33% and recall 70.99%.

4.2.3 At-Least-One vs. Multiple Correct Predictions

One way to evaluate the effectiveness of our approach is to ask the following questions.

– *How well does our methodology predict at least one hashtag correctly?* This is answered by examining whether there is any overlap between the recommended hashtags for the tweet and the ground truth (actual hashtags seen in the tweet). In the joint distribution maximization based recommendation approach, we observe at least one hashtag to have been recommended (predicted) correctly in 66.74% cases.

– *How well does our methodology predict more than one hashtag correctly?* This is answered by examining whether at least two (or more) hashtags overlap, between the recommended hashtags for the tweet and the ground truth (actual hashtags

seen in the tweet). In the joint distribution maximization based recommendation approach, we observe two or more hashtags to have been recommended (predicted) correctly in 42.24% cases.

4.2.4 Comparison with Other Works

In absence of benchmark datasets for comparison, we create a LDA-based baseline score. For this, akin to the rest of our approach, we pick the top 3 topics that the test tweet is aligned to. For each topic, we pick the one representative tweet that has the highest likelihood of belonging to that topic (amongst all the tweets that represent the topic). We perform hashtag assignment to the test topic, using the 3 training tweets thus selected across the 3 topics. The LDA baseline gives 7.79% F1-score. Since our system yields a best-case F1 performance of **46.59%** (with the semantic similarity based approach), the lift we obtain over the LDA baseline is $46.59/7.79 \approx$ **6**, which is large.

Method	Lift
Naive Bayes	3.27
IBM1 (Liu et al., 2011)	3.55
TopicWA (Ding et al., 2012)	4.71
TTM (Ding et al., 2013)	5.87
SemTagger (Joint Maximization)	**5.94**
SemTagger (Semantic Similarity)	**6**
CNN+Att.-5 (Gong and Zhang, 2016)	6.42

Table 3: Lifts over the baseline, across methods

Further, we observe that, our model (F-score 46.59) yields an F-score higher than the literature (39.8). However, in absence of benchmark data, we compare our work with the literature using the **lift** over the baseline LDA values. Table 3 captures these values. Clearly, the lift obtained by our work is comparable to Gong and Zhang (2016), and it consistently outperforms the rest of the literature.

5 Discussion

We discuss a few interesting observations below.

5.1 Significance of Using Lift as a Measure

No standard dataset has been made available yet for the task of hashtag recommendation. Further, many of the existing literature have not released codes, and reimplementation of these codes are always prone to errors. We note that, the methodology that has acted as the benchmark of baseline, is the LDA-based approach. Given these observations, we use the list obtained by the model over baseline LDA, as the approach for validation. Here, the well-known LDA baseline is implemented. Subsequently, a ratio of the performance (F-score) obtained by our system, is compared with that obtained by the baseline LDA implementation. Further, comparing the performance of our system with other works in the literature, becomes meaningful and error-free by this comparison mechanism, in spite of absence of benchmark data as well as released codes for the task of Twitter hashtag recommendation.

5.2 General Observations

Our model is highly novel, and the lift we obtain (lift ≈ 6) is comparable to the state-of-the-art (Gong and Zhang, 2016), and it outperforms all other works available in the literature. Further, our model is lightweight and robust, as opposed to the computationally expensive deep-learning approach of the state-of-the-art. This makes our work useful and effective in real-life applications. We also note that the difference of performance between the two models we proposed - the *semantic similarity based* model and the *joint optimization based model* - is not much, though, the former model performs marginally better compared to the later for the current dataset.

6 Conclusion

In this paper, we proposed a novel hashtag recommendation approach for tweets, based on semantic similarity. We used LDA-based topic model training. For assigning hashtags to a target tweet, we proposed two variants. In one variant, hashtags are assigned to a target tweet, such that, the hashtags are obtained from a set of representative training tweets having the highest semantic similarities with the target tweet. In the other variant, we assigned hashtags to target tweets using (a) a maximization function that combines the probability of a given target tweet belonging to a topic, and the semantic similarity of representative training tweets that belong to that topic, and (b) assigning hashtags observed in the top-K ranked tweets in the maximized combination. Empirically, our model produced a major lift of 6 times over the LDA baseline. Our approach is robust and lightweight, and usable in real-life settings. *SemTagger*, our propose model, will be useful in applications that recommend hashtags to users, for

assigning to tweets and other social network posts, while they post text content on social network platforms, and also, can be used in other social network based applications.

References

Sitaram Asur, Bernardo A Huberman, Gabor Szabo, and Chunyan Wang. 2011. Trends in social media: Persistence and decay. In *ICWSM*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd international conference on computational linguistics: posters*, pages 241–249. Association for Computational Linguistics.

Kuntal Dey, Ritvik Shrivastava, and Saroj Kaushik. 2016. A paraphrase and semantic similarity detection system for user generated short-text content on microblogs. In *COLING*, pages 2880–2890.

Kuntal Dey, Saroj Kaushik, and L Venkata Subramaniam. 2017. Literature survey on interplay of topics, information diffusion and connections on social networks. *arXiv preprint arXiv:1706.00921*.

Zhuoye Ding, Zhuoye Zhang, and XuanJing Huang. 2012. Automatic hashtag recommendation for microblogs using topic-specific translation model. In *24th International Conference on Computational Linguistics*, page 265.

Zhuoye Ding, Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2013. Learning topical translation model for microblog hashtag suggestion. In *IJCAI*, pages 2078–2084.

Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. 2013. Using topic models for twitter hashtag recommendation. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 593–596. ACM.

Yuyun Gong and Qi Zhang. 2016. Hashtag recommendation using attention-based convolutional neural network. In *IJCAI*, pages 2782–2788.

Yeyun Gong, Qi Zhang, and Xuanjing Huang. 2015. Hashtag recommendation using dirichlet process mixture models incorporating types of hashtags. In *EMNLP*, pages 401–410.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Lichan Hong, Gregorio Convertino, and Ed H Chi. 2011. Language matters in twitter: A large scale study. In *ICWSM*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Zhiyuan Liu, Xinxiong Chen, and Maosong Sun. 2011. A simple word trigger method for social tag suggestion. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1577–1588. Association for Computational Linguistics.

Zhi Liu, Chen Liang, and Maosong Sun. 2012. Topical word trigger model for keyphrase extraction. In *In Proceedings of COLING*. Citeseer.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit.

Martin F Porter. 2001. Snowball: A language for stemming algorithms.

Surendra Sedhai and Aixin Sun. 2014. Hashtag recommendation for hyperlinked tweets. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 831–834. ACM.

Jieying She and Lei Chen. 2014. Tomoha: Topic model-based hashtag recommendation on twitter. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 371–372. ACM.

Kate Starbird and Leysia Palen. 2012. (how) will the revolution be retweeted?: information diffusion and the 2011 egyptian uprising. In *Proceedings of the acm 2012 conference on computer supported cooperative work*, pages 7–16. ACM.

Oren Tsur and Ari Rappoport. 2012. What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 643–652. ACM.

Jason Weston, Sumit Chopra, and Keith Adams. 2014. # tagspace: Semantic embeddings from hashtags.

Eva Zangerle, Wolfgang Gassler, and Gunther Specht. 2011. Recommending#-tags in twitter. In *Proceedings of the Workshop on Semantic Adaptive Social Web (SASWeb 2011). CEUR Workshop Proceedings*, volume 730, pages 67–78.

Qi Zhang, Yeyun Gong, Xuyang Sun, and Xuanjing Huang. 2014. Time-aware personalized hashtag recommendation on social media. In *COLING*, pages 203–212.

Algorithm 1 THE SEMTAGGER ALGORITHM

function **CleanAndPreprocess ():**
$t'_r \leftarrow$ Raw tweets posted by user on Twitter
$t'_r \leftarrow t'_r - t'_{\tilde{h}}$, *i.e.*, remove all tweets without any hashtag
$t'_r \leftarrow t'_r - t'_{\tilde{en}}$, *i.e.*, remove all tweets not in English
$t'_r \leftarrow t'_r -$ char(non-ascii), *i.e.*, remove all non-ASCII characters
$t'_r \leftarrow$ norm(t'_r): perform tweet normalization using net-slang and Han-Baldwin
$t_r \leftarrow$ stopword_remove(t'_r): remove stopwords
return $T \leftarrow \{t_1, t2, ..., t_r, ..., t_n\}$, the cleaned and preprocessed tweets

function **LDABasedTopicModeling (Tweets T):**
$H \leftarrow \{h_1, h_2, ..., h_m\}$: set of hashtags present in the training set
$D \leftarrow \bigcup_{i=1}^{n}(t_i) - \bigcup_{j=1}^{m}(h_j)$: concatenation of all tweets, minus all the hashtags
$Z \leftarrow \{z_1, z_2, ..., z_l\} \leftarrow LDA(D)$: the set of topics identified to be present in the document D
return Z, a set of topics learned over LDA

function **RepresentativeTweetIdentification (Tweets T, Topics Z, Top-Ranks R as Integer):**
for $z_l \in Z$ **do**
    for $t_i \in T$ **do**
        $p_{t_i,z_l} \leftarrow$ LDA-based probability of tweet t_i to belong to topic z_l
        $t'_{z_l} \leftarrow$ insert t_i in sorted order of the value of p_{t_i,z_l}
    end for
    $t_{z_l} \leftarrow$, retain the highest R values contained in t'_{z_l}, discard the rest
end for
return $T_{z,L} \leftarrow \{t_{z_l}\} \forall (l \in L)$

function **SemanticSimBasedRec (Target Tweet t, Topics Z, R representative tweets T_{Z_q} across all topics Z_q, Integer K):**
Find the probability p_l of target tweet t to belong to each topic $z_l \in Z$
Sort by p_l and retain Z_q, the top Q topics
for all retained topics Z_q **do**
    $SS'(t, t_{Z_q}) \forall (t_{Z_q} \in T_{Z_q}) \leftarrow$ semantic similarity of target tweet t with representative tweet t_{Z_q}
end for
$SS \leftarrow$ Sort($SS'(t, t_{Z_q})$)
return Hashtags present in the top-K ranked tweets in SS

function **JointDistrMaxBasedRec (Target Tweet t, Topics Z, Integer K):**
for all topics z_l **do**
    for all representative tweets t_j in topic z_l **do**
        $SS(t, t_j) \leftarrow$ semantic similarity of target tweet t with representative tweet t_j
        $P(t, z_l) \leftarrow$ the LDA-based probability p_{t,z_l} of target tweet t to belong to topic $z_l \in Z$
        $CS'(t, t_j) \leftarrow SS(t, t_j) \times P(t, z_l)$
    end for
end for
$CS \leftarrow$ Sort($CS'(t, t_j)$)
return Hashtags present in the top-K ranked tweets in CS

Reasoning with Sets to Solve Simple Word Problems Automatically

Sowmya S Sundaram
Indian Institute of Technology, Madras
Chennai 600036
sowmya@cse.iitm.ac.in

Deepak Khemani
Indian Institute of Technology, Mandi
Himachal Pradesh 175005
khemani@iitmandi.ac.in

Abstract

A system, Magi, is proposed, which analyses simple addition/subtraction arithmetic word problems expressed in English, represents them in the form of schemas and sets, reasons with set cardinalities and presents the final answer in English phrases. It also provides simple explanations. This work presents a study of the features of a knowledge-based system used for solving such a task. It has been evaluated and has been found to perform better than current knowledge-based systems for similar problems.

1 Introduction

Natural language understanding is one of the key elements of human intelligence. Hence, it has attracted the attention of a sizeable population of researchers of artificial intelligence. The first published work in this field (Bobrow, 1964) attempted to solve word problems presented to a computer in English. The appeal of solving word problems lies in the fact that semantic understanding is required to map the word problem to a mathematical framework.

Consider the following example.
Input: Keith has 20 books . Jason has 21 books . How many books do they have together?
Output: Altogether 41 books

Here, the system has to map the word 'they' to 'Keith' and 'Jason'. This is an example of co-reference resolution. Next, the notion of associating '20 books' to 'Keith' and '21 books' to 'Jason' has to be captured. These relevant details are also extracted. The last piece of information required is the word 'together' that signifies what is the goal of the problem. In this work, these details are extracted by using the Stanford Core NLP (Manning et al., 2014) suite of tools extensively. Other approaches include semantic parsing (Shi et al., 2015), learning equation co-efficients (Kushman et al., 2014), learning expression trees (Koncel-Kedziorski et al., 2015) and so on. In this work, as the principle was to build as precise a system as possible, we've used a rule based approach. The motivation was that if this tool was used by a student, she should be able to see the trace of the solution.

In order to know what are the elements that are to be extracted from the word problem, some model of word problems must be encoded into the system. This is the role of knowledge representation. Here, knowledge representation is in the form of schemas that are templates for solving problems. They describe common categories of word problems. The schema for the above problem is 'combine' which describes that the answer is the sum of entities in question. Internally, this idea is represented as sets for closer coupling to the semantics of the problem.

t_0
Jason has B books
Keith has A books
———————
A 20
B 21

The next step is reasoning. The schema 'combine' directs that the sum of the 'books' owned by 'Jason' and 'Keith' is required. The answer is computed by adding the cardinalities of A and B. The final answer, '41' is presented as 'Altogether 41 books'. The last step of generating the answer is facilitated by the schema as well.

The challenges in this problem solving process are high. This is because natural language processing is difficult and often ambiguous or

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 188–196

may rely on implicit details. Magi resolves some of the ambiguities by reasoning about implicit events and making some assumptions. There are some ambiguities in schema identification as well which have been partially addressed using some heuristics. While computing elementary problems, numerical efficiency of computers is much more. The challenging task is the introduction of language and representing the information extracted from the natural language.

2 Related Work

As mentioned, the work that pioneered the field of natural language understanding was (Bobrow, 1964). The program, STUDENT, was able to process sentences that followed a specified template and could handle addition, subtraction, division, multiplication and exponentiation.

Schemas were introduced by (Fletcher, 1985) based on cognitive theory. It specified three schemas - combine, compare and change. After this, (Dellarosa, 1986) proposed ARITHPRO which encoded some inheritance. For example, dolls and balls are toys. If a problem described dolls, balls and clothes and the task was to find total toys, ARITHPRO would pick only relevant entities.

Schemas were used relatively recently in (Bakman, 2007). It could solve multi-step problems and could ignore extraneous information. However, the system did not scale well as the complexity of natural language increased. This is a common trait running through all these knowledge-based systems. A major stumbling block was the complexity of natural language understanding. Many systems worked on a subset of natural language called Controlled Natural Language to resolve ambiguities.

After this system, as mentioned in (Mukherjee and Garain, 2008), without a common standard dataset to compare different algorithms, the interest in this field died down. Also, the extensive human effort involved in curating rules for these systems was not encouraging.

In recent times, there has been a resurgence of interest for this type of problems. In (Kushman et al., 2014), word problems were solved by building an empirical model that matched the numerals in a word problem to co-efficients in a template. Their domain was the set of word problems that could be solved by a set of linear equations. They achieved a commendable accuracy. (Zhou et al., 2015) improved this work by using quadratic programming on a simpler and more efficient model. Another work (Hosseini et al., 2014) used a state representation for arithmetic word problems. It used machine learning to identify the characteristic operation signalled by a verb. They provided three datasets of varying difficulty that have been used for evaluation in this paper against their fully knowledge-based variant of the code. (Shi et al., 2015) solves algebraic word problems which reason about numbers and their relations. It uses semantic parsing with a custom-built language for their chosen domain. The work presented in (Koncel-Kedziorski et al., 2015) learns a model that maps natural language to expression trees. They could solve single-variable word problems effectively. As the narrative of the problem became longer, the search space grew exponentially. A knowledge based system could potentially solve problems of arbitrary length provided the sentences could be processed by it. In (Mitra and Baral, 2016), there is a notion of the categories of word problems where a model learns to identify which category and the alignment of the numbers to the template of the word problem. It is the learning version of this work. The drawback it faces is the heavy annotation required for learning such an alignment. A recent work, (Ling et al., 2017) uses deep learning to solve general word problems and provide explanations for the same. It solves problems which are in a competitive exam style, with possible answer options. It develops a language model and a mathematical model simultaneously. Since the problem setting is slightly different, its performance on existing datasets is not available.

The knowledge based systems are precise but can attempt few real-world problems as their natural language processing is limited. On the other hand, empirical systems can tackle a wide gamut of problems but they are not as precise as knowledge based systems. This work hopes to maximise the trade-off between the two methodologies by using statistically trained parsers and a well-defined representation system.

3 The Process

As the dependency parser provided by (Manning et al., 2014) is not robust for long sentences, the given English word problem is first simplified and

then passed to the co-reference resolver and the dependency parser. The simplification is based on a set of rules derived from the part-of-speech tags and the constituency parser. The parsers extract relevant information for each sentence. The sentences are then ordered in increasing order of time by using the tense of the sentences. After this, sets are created for each numerical entity. Then, relationships between these sets are established by schemas using the extracted information. Finally, after reasoning with the set cardinalities, the answer is displayed along with the trace and explanation. This process is explained in detail with a running example

Algorithm 1: The Problem Solving Process

```
Input: Word Problem: p
Output: String: expl, Number: ans
WordProblem p1 = simplify(p)
List<Steps> extractedInfo =
  extract(p1.sentences())
List<Steps> orderedInfo =
  rearrange(extractedInfo)
time = 0
questionSet = ∅
story = ∅
for each step in orderedInfo do
    if event(step) then
        time = time + 1
    end
    story = story.add(step,time)
    story = story.apply(step.schema)
    if step.isQuestion = true then
        questionSet =
          story.get(step,time).value
    end
end
solve(story.sets)
expln = explain(story)
ans = questionSet.cardinality
Print ans
Print expln
```

4 Natural Language Understanding

The steps involved in natural language understanding are explained briefly below.

- Resolve unknown entities - For example, if the problem had sentences like, 'There are 5 trees in a park. Park workers cut 2 of them', this is converted to 'There are 5 trees in a park. Park workers cut 2 trees.' This is to ease the task of the parsers.

- Simplifying sentences - Most long sentences are split into simpler sentences. For instance, 'Sally got 4 erasers and 3 pencils' is split as 'Sally got 4 erasers. Sally got 3 pencils.'

- Resolving co-references - This has been done by using the 'decoref' annotator provided by (Manning et al., 2014).

- Rule based information extraction - a set of rules that work on the output given by the dependency parser to extract the details of each sentence .

- The retrieved information is then ordered based on the tense of each sentence.

4.1 An Example

Let us see an example to illustrate the various points described above. Consider the problem, 'Molly owns the Wafting Pie Company. This morning, her employees used 816 eggs to bake pumpkin pies. If her employees used a total of 1339 eggs today, how many eggs did they use in the afternoon?'.

4.1.1 Preprocessing

For this problem, the first step is to resolve the pronoun 'her'. After this step, our system changes the input to, 'Molly owns the Wafting Pie Company. This morning, *Molly's* employees used 816 eggs to bake pumpkin pies. If *Molly's* employees used a total of 1339 eggs today, how many eggs did *Molly's employees* use in the afternoon?'.

4.1.2 Simplification

As the sentences are relatively complex, simplifying them brings much better results. At the end of simplification, the problem is changed to : 'Molly owns the Wafting Pie Company. Molly's employees used 816 eggs to bake pumpkin pies. Molly's employees used a total of 1339 eggs today. How many eggs did Molly's employees use in the afternoon?'. This is achieved by examining the Part-Of-Speech(POS) tag of every sentence, identifying the verb, and extracting the relevant noun phrase and verb phrase.

4.1.3 Information Extraction

Some rules are used to get the information required for representation. Each sentence's analysis is enlisted below.

- 'Molly owns the Wafting Pie Company' - this sentence is ignored because there is no number involved. There are some exceptions to this rule. If the sentence contains words like 'some', that information is encoded.

- 'Molly's employees used 816 eggs to bake pumpkin pies' - this sentence is converted by Magi as

owner1	:	Molly's employees
owner2	:	(none)
verb	:	use
entity	:	egg
value	:	816
keyword	:	use
procedure	:	reduction
tense	:	past
isQuestion	:	false
isAggregator	:	false

  The 'owner1' and 'owner2' fields suggest who are the participants. Here, only 'Molly's employees' are relevant. If the question was 'Sally gave 4 kites to Sam', then the two owners would be 'Sally' and 'Sam' respectively. This is extracted by a set of rules. In this case, the subject of sentence(denoted by the 'nsubj' tag) is taken as the first owner. There are a set of keywords and their associated procedures stored - this is explained in more detail in the next section. If the sentence contains one of the keywords, it is retrieved from the sentence and stored along with the corresponding procedure's name. The tense is stored by analysing the POS tag. It is later used for ordering. The field 'isQuestion' signifies whether this step contains information that pertains to the answer to be retrieved. On the other hand, 'isAggregator' states whether the sentence contains any word that imply combination, such as 'total', 'altogether', etc.

- 'Molly's employees used a total of 1339 eggs today.' - a similar process leads to the following data to be stored.

owner1	:	Molly's employees
owner2	:	(none)
verb	:	use
entity	:	egg
value	:	1339
keyword	:	use
procedure	:	reduction
tense	:	past
isQuestion	:	false
isAggregator	:	true

- 'How many eggs did Molly's employees use in the afternoon?' -

owner1	:	Molly's employees
owner2	:	(none)
verb	:	use
entity	:	egg
value	:	(empty)
keyword	:	use
procedure	:	reduction
tense	:	past
isQuestion	:	true
isAggregator	:	false

  By setting the 'isQuestion' flag, the system is now equipped with the insight that the answer required is the number of eggs Molly's employees used.

5 Knowledge Representation

5.1 Schemas

Schemas are templates that suggest how a problem should be solved. They were applied to solve math word problems first by (Fletcher, 1985). He used three schemas - Combine, Change and Compare. Let us consider the 'Compare' schema. A typical example is 'Rachel has 3 pencils. Tom has 3 pencils more than Rachel. How many pencils does Tom have?'. The schema, 'Compare', and its instantiation is given below:
(owner1) has (X) (object)
(owner2) has (Y) (object) more than (owner1)
(owner2) has (Z) (object)
Z = X + Y
(owner1) = 'Rachel', (owner2) = 'Tom', (object) = 'pencil', (X) = 3, (Y) = 3.

Schemas have a structure that can be mapped to the sentences given in the sentence along with an

equation connecting the variables. If two variables are retrieved from the problem (in this example, X and Y), the value of the third variable can be computed.

The first disadvantage of this method is that it is too rigid. All word problems are not expressed in a format that is easy to map to this format. If the question was changed as 'Rachel has 3 pencils. Tom has 3 more. How many does he have?', this particular schema would fail as it does not exactly match the natural language input expected. The second issue is that these three schemas are inadequate to describe all types of problems. For example, the problem 'Samantha has 8 cookies. She ate 3 of them. How many does she have now?', would not fit in any of the above schemas. The 'change' schema is tailored to capture transfer of 'object' from one person to another. Hence, though it seems applicable, it is not so.

To counter these challenges, (Bakman, 2007) describes his system 'ROBUST' that can handle a larger number of schemas. Some ideas were inspired by Script Applier Mechanism (SAM) by (Schank and Abelson, 1975) which captured semantics by grouping words from a dictionary into categories. Similarly, instead of a single keyword for schemas, ROBUST mapped a lot of keywords to a single schema. For example, 'eat', 'destroy', 'kill' were keywords for the 'termination' schema. ROBUST concentrated on the various possibilities of 'change' schema. It also used schemas iteratively until all possible equations were applied in order to handle multi-step problems. It showed significant improvements over existing systems.

5.2 Schemas and Time

From ROBUST's emphasis on the 'change' schema, the next natural step is to capture information about time. By explicitly assigning timestamps to sentences, the search for schema instantiation is made more focussed.

5.3 Schemas and Ambiguity

A classic example of ambiguity can be seen in the problem 'Samantha ate 8 cookies. Anne ate 4 cookies more than Samantha. How many cookies did Anne eat?'. Here, the correct schema to be used is 'Compare'. However, due to the word 'ate', the 'termination' schema is also applicable. If the 'termination' schema is applied, since there is no information about the cookies any of them had before or after, it cannot be instantiated. To address this, the schemas are modified such that the verb is variable and it can reason about any verb. The narrative is represented in the following manner.

t_0
Samantha : eat : 8 : cookies
t_1
Anne : eat : 8 + 4 : cookies

Hence, the template-matching is relaxed and more problems can be solved.

5.4 Schemas and Sets

After introducing time, its related concepts and reasoning about events while applying schemas, there are still some problems which cannot be addressed. Consider, 'There are 70 students in a class. If 65 students are present, how many are absent?'.

These problems have no events, or tell-tale keywords for helping the system solve problems. The schema of combination usually implies an aggregation over different owners. This a case of set completion, where the 65 students are a subset of the 70 students in class and the set of students who are present is disjoint from the set of absentees. Hence, the representation is shifted to schemas with descriptions as sets.

Revisiting 'Rachel has 3 pencils. Tom has 3 pencils more than Rachel. How many pencils does Tom have?', the 'compare' schema which has been specialised as 'compare-plus' in Magi is stored as :
(owner1) (verb) (X) (object)
(owner2) (verb) (Y) (object) more than (owner1)
(owner2) (verb) (Z) (object)
$|Z| = |X \cup Y|, X \cap Y = \emptyset$
(owner1) = 'Rachel', (owner2) = 'Tom', (verb) = 'has', (object) = pencil, $|X| = 3, |Y| = 3$.

Coming back to the set-completion scenario, the narrative is represented as :
t_0
class : has : A : students
class : has : B : present students

A 70
B 65

While parsing the sentence, the behaviour of

antonyms is recorded and the case for subset completion is set to be true, if the antonyms are appropriately situated. If it is true, the statements $B \subseteq A, C \subseteq A, B \cap C = \emptyset$ are added along with 'class : has : C : absent students'. Antonyms have been computed from `https://www.thesarus.com`.

5.5 Magi's Schemas

The schemas used by Magi are described in Table 1. The procedures are programming directives and are more flexible than traditional schemas. Implicitly, all sets are considered disjoint unless set completion is involved. Even though the description says 'owner1' and 'owner2', Magi is implemented in such a way that it can reason about different entities owned by the same owner if required.

Schema	Procedure	Relations
combine	Sum over all relevant entities	$\|D\| = \|A \cup B..\|$
comparePlus	owner1 has A items, owner2 has B items more, owner2 has C items	$\|C\| = \|A \cup B\|$
compareMinus	owner1 has A items, owner2 has B items less, owner 2 has C items	$\|C\| = \|A - B\|$
increase	owner1 had A items, owner1 got B items more, owner1 has C items now	$\|C\| = \|A \cup B\|$
reduction	owner1 had A items, owner1 lost B items, owner1 has C items now	$\|C\| = \|A - B\|$
set-completion	A,B,C	$B \subseteq A, C \subseteq A$

Table 1: Flexible Schemas used by Magi

6 Reasoning

In the straight-forward situation, reasoning is simply a case of solving the equations relating set cardinalities based on the axioms of set theory. However, to address a larger type of problems, some common sense rules have been added.

6.1 Handling Implicit Events

Consider the problem, 'Last week Tom had $74. He washed cars over the weekend and now has $86. How much money did he make washing cars?'. The word 'wash' is not a keyword, hence it is not registered as an event. When the narrative is being constructed, the first statement will record that Tom has 74 dollars. As no event has occurred, the timer is not incremented. After that, the system encounters that Tom has 86 dollars at the same time. Since this is not possible, it introduces an event, 'Tom gets 86 - 74 dollars'. This is illustrated below:

t_0
Tom : has : A : dollars
t_1
Tom : get : C : dollars
t_2
Tom : has : B : dollars

A 74
B 86
C 12

The statements involved are $|C| = |B| - |A|$.

6.2 Assumption of Initial Conditions

Most schema-based systems fail due to some missing information. For example, ROBUST would fail to solve 'Jane bought 10 cookies. She ate 3 cookies. How many does she have now?'. This is because, it would try to find some value as the initial number of cookies Jane owned. Magi sets initial values as $\emptyset$.

6.3 Reasoning about Events

Ideas from 'Event Calculus' described in (Shanahan, 1999) have been used to construct the narrative. For example, circumscription is used in the problem, 'Sam grew 4 watermelons, but the rabbits ate 3 watermelons. How many watermelons does Sam have?' to reason that the 3 watermelons are actually a subset of Sam's watermelons. This idea was also employed by (Hosseini et al., 2014). Also, common sense law of inertia was implemented to state that entities that were not affected by an event, continue to persist across time steps.

6.4 Reasoning and Natural Language

Sometimes, reasoning is performed using the extracted information presented for representation. For example, the situation where 'Sam buy games for $35' actually implies that the event is 'Sam spent 35 dollars'. Such rules are also enforced.

6.5 Heuristics

Due to the complexity of processing natural language and the limited rules available, the numerals in the problem may not be correctly assigned to the templates required for a schema properly. Hence, some heuristics are used to improve performance. As expected, they are not sound. One consistent heuristic is, if Magi

retrieves a value that is already given in the question, then search is repeated. Another heuristic is, if the system is unable to represent as desired, but has recognized that the question needs aggregation, it simply returns the sum of all entities.

7 Natural Language Generation

Since one of the driving factors behind this work is to facilitate the understanding of students, an attempt has been made to explain the answer obtained in natural language. The trace of the problem is recorded as the problem is solved and then an explanation is generated. The quality of generation is quite low at this point in time but the intermediate representation is provided. This can be taken up by a generation expert and designed.

One of the successful examples is illustrated below.

7.1 Problem

John had 7 apples. Mary has gave some apples to John. Now, John has 10 apples. How many apples does Mary have?

7.2 Explanation

John has 7 apples.
Mary gives x apples.
Hence, John has 7+x apples.
Now, John has 10 apples.
Therefore, 7+x = 10
Mary gives 3 apples

7.3 Trace

John had 7 apples. Mary gave 3 apples to John. John had 10 apples. John had 10 apples. Mary had 3 apples.

This trace is concatenating the situation at every time step. Hence, the statement John had 10 apples is repeated twice. We used SimpleNLG (Gatt and Reiter, 2009) for generation.

8 Evaluation

Magi has been coded in Java 1.7 and has used the same version of (Manning et al., 2014) parser as the one used by ARIS (Hosseini et al., 2014) for a fair evaluation. The work has been compared against other knowledge based systems.

Magi has been evaluated on the three datasets, DS1, DS2 and DS3 provided by (Hosseini et al., 2014). DS1 has 134 problems. DS1 and DS3 are similar in terms of the applicable schemas. However, DS3 has more complex sentences and has extraneous information. It has 121 problems. DS2 involves the use of decimals which is difficult for parsing. Also, DS2 has a lot of problems that require set-completion, an issue whose solution was the inspiration for this representation. It has 140 problems.

	DS1	DS2	DS3	Avg
Magi	**95.52**	**80.00**	**84.30**	**86.51**
Gold ARIS	94.0	77.1	81.0	84.0
ROBUST	12.69	0.71	0	4.56
WolframAlpha	5.97	2.14	0.83	3.03

Table 2: Performance

A comparison has been presented in Table 2 with respect to three other systems. One is ROBUST (Bakman, 2007) which has been discussed before. (Hosseini et al., 2014) presented ARIS. It attempted to learn the equation categorising verbs. They also presented an algorithm for learning that information. However, by limiting themselves to verbs (change schemas), other schemas such as 'combine' and 'compare' were missed. As we have not performed any empirical method to learn the keyword-schema mapping, the system for comparison is Gold-ARIS. (Wolfram, 2015) is another system that solves math word problems on the Internet without divulging implementation details.

The increased performance over Gold ARIS is because of the use of heuristics, addressing set completion and handling implicit events. Also, simplifying the problem and performing some reasoning with the language helped reduce parser errors mentioned in (Hosseini et al., 2014). ROBUST performs relatively better with DS1 because it consists of simple sentences. As the complexity of processing English increases, the performance of both ROBUST and WolframAlpha reduces.

8.1 Analysis of Errors

8.1.1 Absence of a Keyword

Consider "A restaurant served 9 pizzas during lunch and 6 during dinner today. How many pizzas were served today?". Since there are no keywords like "altogether", the system did not recognize that the "combine" schema is to be applied. Also, it could not identify that lunch and dinner are parts

of "today".

8.1.2 No Model for Intent

In "Joan decided to sell all of her old books. She gathered up 33 books to sell. She sold 26 books in a yard sale. How many books does Joan now have?", the system couldn't represent that Joan hadn't actually sold 33 books and was only intending to sell them.

8.1.3 Issues in Extracting Entities

Consider "A ship full of grain crashes into a coral reef. By the time the ship is fixed, 49952 tons of grain have spilled into the water. Only 918 tons of grain remain onboard. How many tons of grain did the ship originally contain?". The system did not recognize that the ship had spilled tons of grain. The system represented it as "water has 49952 ton" and "water spill 918 ton". Here there are multiple entities that are interacting with each other. These facets could not be extracted by the rules designed for information extraction.

9 Discussion

While it has been presented that Magi is a good knowledge-based system, the question remains whether it is robust enough to have a recall comparable with empirical systems. This is hard to evaluate as empirical systems are usually tested by cross validation. When a human expert makes rules, she cannot subjectively claim that the rules have made solely on the basis of a section of the data. The fact that the system can achieve a high accuracy shows that it does solve a large number of problems. However, there may be a problem of over-fitting in some sense. In empirical systems, this is also possible because often there is a considerable overlap of sentence styles in the training and test examples. How these systems fare with completely unseen data would be an interesting experiment to compare these algorithms. This work is not limited to presenting a numerical answer. Rather, it attempts to illustrate what are the components required to build a product that would benefit students - namely natural language understanding, representation, reasoning and natural language generation. A loss in precision implies that it might induce confusion in a student's mind. In hindsight, the heuristics did drastically reduce precision and doing away with them is part of the future work.

9.1 Knowledge Acquisition Bottleneck

The primary reason knowledge-based systems went out of vogue for natural language processing is because of the knowledge acquisition bottleneck. In this work, the types of word problems were already established in the literature. However, two sources of knowledge acquisition bottleneck still exist - the mapping between the schemas and the keywords as well as the various rules and strategies to extract information and represent them as schemas. While a human expert can sift through the data and come up with better rules than a learning program in a simplistic domain such as this, the generalisability of this approach is questionable.

9.2 Similarities between Knowledge-Based and Empirical Systems

In this particular domain, there is often a need to encode world knowledge in some form. In empirical systems, it comes as the cost of annotation and choice of hand-coded features.

9.3 The Need for Semantics

Many works (eg. (Shi et al., 2015)) recognize the need for semantics for this class of problems. A single word could completely change the equation construction. Hence, it is imperative that there must be some model of mathematical computation.

10 Conclusion and Future Work

We have presented a knowledge-based system to explore what are the exact sources of information required in the quest for building a student-friendly application that is precise. It has been shown that to solve such problems, world knowledge has to be encoded and semantic understanding is required. Exciting developments such as deep learning (Ling et al., 2017) in natural language processing can learn the required representation as well and succeed in building an end-to-end system. However, it comes at the cost of a huge amount of data that is not available at this point in time for many mathematical problem domains. Though we have introduced some level of statistical analysis through parsers, it would be beneficial to explore semantic parsing and other approaches to map the natural language description to an underlying representation with higher precision for semantically richer domains.

References

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. *arXiv preprint math/0701393*.

Daniel G Bobrow. 1964. A question-answering system for high school algebra word problems. In *Proceedings of the October 27-29, 1964, fall joint computer conference, part I*, pages 591–614. ACM.

Denise Dellarosa. 1986. A computer simulation of childrens arithmetic word-problem solving. *Behavior Research Methods, Instruments, & Computers*, 18(2):147–154.

Charles R Fletcher. 1985. Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers*, 17(5):565–571.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93. Association for Computational Linguistics.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. *ACL (1)*, pages 271–281.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. ACL.

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2):93–122.

Roger C Schank and Robert P Abelson. 1975. *Scripts, plans, and knowledge*. Yale University.

Murray Shanahan. 1999. The event calculus explained. In *Artificial intelligence today*, pages 409–430. Springer.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal*.

Stephen Wolfram. 2015. Wolfram—alpha. *On the WWW. URL http://www. wolframalpha. com*.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *EMNLP*, pages 817–822.

Improving NER for Clinical Texts by Ensemble Approach using Segment Representations

Hamada A. Nayel
Department of Computer Science
Benha University, Benha-Egypt
Mangalore University, Mangalore-India
hamada.ali@fci.bu.edu.eg

H. L. Shashirekha
Department of Computer Science
Mangalore University,
Mangalore-574199, India
hlsrekha@gmail.com

Abstract

Clinical Named Entity Recognition (Clinical-NER), which aims at identifying and classifying clinical named entities into predefined categories, is a critical pre-processing task in health information systems. Different machine learning approaches have been used to extract and classify clinical named entities. Each approach has its own strength as well as weakness when considered individually. Ensemble technique uses the strength of one approach to overcome the weakness of another approach by combining the outputs of different classifiers in order to make the decision thereby improving the results. Segment representation is a technique that is used to add a tag for each token in a given text. In this paper, we propose an ensemble approach to combine the outputs of four different base classifiers in two different ways, namely, majority voting and stacking. We have used support vector machines to train the base classifiers with different segment representation models namely IOB2, IOE2, IOBE and IOBES. The proposed algorithm is evaluated on a well-known clinical dataset i2b2 2010 corpus and results obtained illustrate that the proposed approach outperforms the performance of each of the base classifiers.

1 Introduction

Named Entity Recognition (NER) is a leading subtask of information extraction originated from the Sixth Understanding Conference (MUC-6) (Grishman and Sundheim, 1996), which aims at identifying Named Entities (NEs) in a text and classifying them into predefined classes. Names of organizations, locations and persons are examples of NEs in general newswire domain, while DNA, RNA and protein are examples of NEs in biological domain. In clinical domain, terms representing problem, treatment and laboratory test are considered as NEs.

The exponential growth of health information systems produce a massive amount of Electronic Health Records (EHRs). It is vital to apply NER for health information systems because EHRs contain NEs representing laboratory test, problem and treatment in unstructured narrative documents (Friedman et al., 1994). Moreover, NER in clinical domain (Clinical-NER) is an important pre-processing task in health information systems where further tasks of health information systems depend essentially on the results of Clinical-NER. Clinical-NER is a challenging problem because, in addition to the general challenges of NER there are other challenges resulting from the nature of clinical NEs such as: -

1. Ambiguity:- the major sources of ambiguity are abbreviations and acronyms (Pakhomov et al., 2005), which are used routinely in clinical texts. Two different cases cause the ambiguity, (i) same abbreviation used for different entities such as "*EF (Ejection Fraction)*" which is used as a medical problem as well as a laboratory test, and (ii) an abbreviation conflicts with a word such as "*VS*" which is a laboratory test as well as abbreviation for the word "*versus*".

2. Multiple words entities:- most of clinical entities consist of multiple words such as "*lower abdominal pain*" and "*chest x-ray*".

3. Nested clinical entities:- some clinical entities occur as a part of longer entity such as

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 197–204

"*BP (blood pressure)*", a laboratory test occurs in "*control BP*" which is a treatment.

4. Polysemy:- same clinical term can represent different meanings based on the context, such as "*inflammation*" may refer to skin problem, a cellular level problem as well as non-medical activity.

5. Synonymy:- a single medical concept can be expressed as multiple words (Dehghan et al., 2013) such as "*baby*" and "*foetus*" which means the same in many medical contexts.

In addition to these challenges, there is no standard nomenclature for clinical entities of same class.

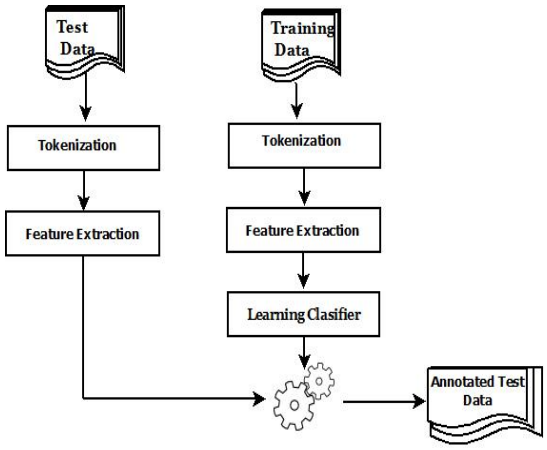


Figure 1: Machine learning framework for NER

1.1 NER approaches

The commonly used approaches for NER are dictionary based approach, rule based approach, Machine Learning (ML) approach and hybrid approach (Keretna et al., 2015). In dictionary based approach, a dictionary or lexicon, which contains a finite set of named entities is used to look up for the entities in texts. Rule based approach uses well-designed domain specific hand crafted rules by experts to match the entities. In ML approach, ML algorithms such as Support Vector Machines (SVMs), Conditional Random Fields (CRFs) and Maximum Entropy (ME) are used to create a learning model using training set to detect the boundaries of entities and classify them into one of the predefined classes. Hybrid approach combines two or more approaches to identify NEs. ML approach either solo or hybrid with another approach is preferable to use as they can easily adopt to new domains as well as identify unseen entities. The major requirement of an ML approach is an annotated data set tagged by experts (*training data*) to train the learning model.

1.2 General Framework of NER using ML approach

Figure 1 shows the general framework of NER using ML approach. In this model, a training data set is used to train the classifier and a set of untagged data (*testing data*) is used to evaluate the performance of the classifier. In tokenization phase, data sets are tokenized into set of tokens or words. In feature extraction phase, a set of features are extracted. Feature extraction is a very important phase as the performance of the model depends essentially on features. Then the features of the training data are used to learn the model and that of testing data are used for evaluation. The success of ML approach depends on the quality of annotated training data, quality of the features extracted as well as the algorithm used for creating the classification model. Each classification algorithm has its strength as well as weakness when used individually. Some classifiers give good results on some datasets whereas the same classifier perform very bad on some other datasets. So, instead of considering a single classifier, it will be beneficially to pool the classifiers and then take the collective decision similar to the decision taken by a committee rather than an individual. This technique which overcomes the weakness of some classifiers using the strength of other classifiers is termed as "ensemble" and is gaining importance for various applications. Ensemble classification uses a set of classifiers preferably weak, diverse and heterogenous classifiers as base classifiers and combines the output of these base classifiers in different ways to get the final output. To achieve the diversity of base classifiers, researchers are using different feature sets, different training sets and/or different classification algorithms. There are different approaches to create ensemble classifiers such as bagging, boosting and stacking (Polikar, 2006). In bagging, different training subsets are drawn with replacement from the entire training data and each training data subset is used to train each base classifier. The outputs of base classifiers are combined using majority voting. Boosting is similar to bagging, but the selection process of training subsets subsequently gives more weight to misclassified samples. Stacking uses outputs

	IO	IOB1	IOB2	IOE1	IOE2	IOBE	IOBES
Treatment	O	O	O	O	O	O	O
/	O	O	O	O	O	O	O
stay	O	O	O	O	O	O	O
IHSS	I-problem	B-problem	B-problem	E-problem	E-problem	B-problem	S-problem
AF	I-problem	B-problem	B-problem	E-problem	E-problem	B-problem	S-problem
ESRD	I-problem	B-problem	B-problem	E-problem	E-problem	B-problem	S-problem
on	O	O	O	O	O	O	O
HD	I-treatment	I-treatment	B-treatment	I-treatment	E-treatment	B-treatment	S-treatment
,	O	O	O	O	O	O	O
IgA	I-problem	I-problem	B-problem	I-problem	I-problem	B-problem	B-problem
nephropathy	I-problem	I-problem	I-problem	I-problem	E-problem	E-problem	E-problem
on	O	O	O	O	O	O	O

Table 1: An example of using different Segment Representation models

of base classifiers to train a new model, which is known as meta-classifier (Wolpert, 1992) and the meta-classifier is used for final classification.

1.3 Segment Representation

Segment Representation (SR) (Cho et al., 2013) involves the process of assigning suitable class label to the words in a given text. SR models have been applied for different tasks such as Part of Speech (PoS) tagging and Noun Phrase chunking (NP-chunking) (Wu, 2014). SR model comprises set of tags, which determine the position of a token in NE, combined with the class label that NE belongs to. The tags used in SR techniques are Begin (**B**), End (**E**), Inside (**I**), Single (**S**) and Outside (**O**). For example, SR for a token is B-XXX means that word is the first word of a NE of class XXX. SR can represent multiple word NEs and nested NEs. Different models are used for segment representation by different researchers. The primary SR model **IO** (Béchet et al., 2000) assigns the tag **I** for the tokens inside the entity and the tag **O** for the tokens outside the entity, but is not able to represent the boundaries of two consecutive entities of the same class. **IOB1** model has been introduced to solve this problem (Ramshaw and Marcus, 1995), by assigning the tag **B** to the first token of consecutive NEs of same class, while **IOB2** model assigns the tag **B** for the first word of each NE (Ratnaparkhi, 1998). **IOE1** and **IOE2** models use same concepts of **IOB1** and **IOB2** respectively, but assigns the tag **E** to the last token of NEs (Kudo and Matsumoto, 2001). Sun et al. (2010) introduced **IOBE** model which concerns with the beginning and end of the NE. **IOBE** model assigns the tags **B** and **E** for the first and last word of all NEs respectively. **IOBES** model is a modified version of **IOBE** model that is concerned with single word NEs. In addition to **IOBE** tags, the **IOBES** model assigns the tag **S** to the NEs of a single word. This model differentiates between the single word and multiple words NEs. Example of tagging the text fragment "*Treatment / stay IHSS AF ESRD on HD , IgA nephropathy on ..*" with different SR models is shown in Table 1.

2 Related Work

The research works carried out in ensemble approach uses different training data sets or different learning algorithms to create the base classifiers. Different ML algorithms such as SVM and CRF have been used for Clinical-NER (Li et al., 2008). Keretna et al. (2014), have introduced a hybrid approach using rule-based and dictionary-based approaches to identify drug names in unstructured and informal texts. The system was evaluated on i2b2 2009 medication challenge dataset and reported 66.97% f-score. Dictionaries and rule-based approaches have been extensively used to extract clinical entities in clinical information systems such as MedLEE developed by Friedman et. al. (1994), MetaMap developed by Arnson and Lang (2010) and cTAKES developed by Savova et al. (2010). Gurulingappa et al. (2010) trained CRFs on textual features enhanced with the output of a rule-based NER system. They evaluated their work using i2b2/VA 2010 medical challenge dataset and reported 81.2% f-measure. Halgrim et al., (2010) designed a hybrid approach that comprised of CRF and Rule-based approach for Clinical-NER. Zhang and Elhadad (2013) de-

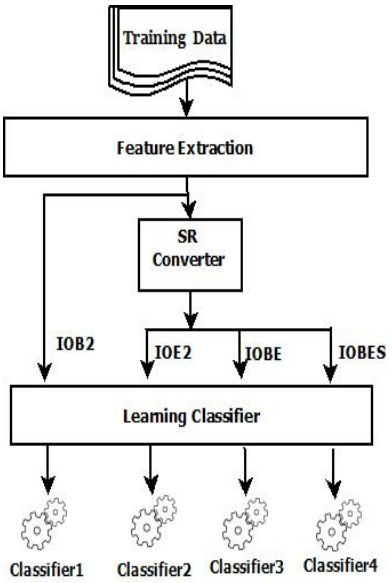


Figure 2: Learning base classifiers

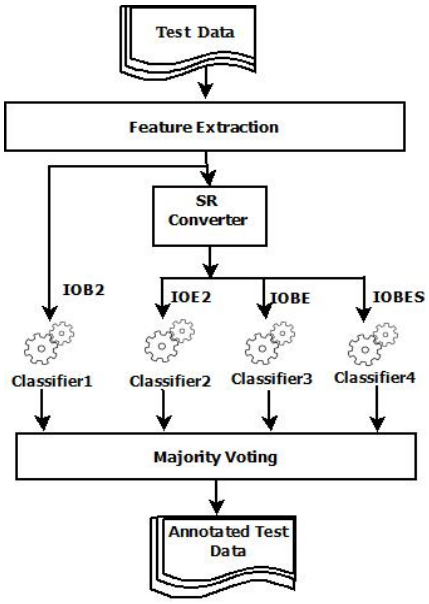


Figure 3: Combining base classifiers using Majority Voting

veloped an unsupervised approach for extracting clinical entities from free text. They used inverse document frequency as a base to filter candidate clinical NEs. Ekbal and Saha (2013) used stacked ensemble approach to extract biomedical NEs. Shashirekha and Nayel (2016) studied the performance of biomedical NER using different SR models. Keretna et al. (2015) introduced a technique for boosting clinical-NER by extending IOBES model, and have introduced a new tag to resolve the problem of ambiguity. They evaluated the proposed technique on i2b2/VA 2010 medical challenge dataset. There is a growing interest in studying Clinical-NER for non-English texts (Wu et al., 2015; Spat et al., 2008). Wu et al. (2015) trained a deep neural network model to extract clinical entities from Chinese texts .

In this paper, we have proposed an ensemble algorithm for Clinical-NER. Up to our knowledge, this is the first work that uses SR models to achieve diversity of base classifiers. Our approach is a two-stage ensemble algorithm. In the first stage, we have used SVM algorithm to create four base classifiers with different SR models namely IOB2, IOE2, IOBE and IOBES. Stacking using CRF as a meta-classifier and Majority Voting have been used separately to combine the results of base classifiers in the second stage.

3 Methodology

We propose a two-stage ensemble approach for clinical-NER. Figure 2 shows the framework of first phase, where training data is used to learn the base classifiers. We have used SVM algorithm to learn four different base classifiers using different SRs models namely, IOB2, IOE2, IOBE and IOBES. In second phase, we have combined the outputs of the base classifiers created in the first phase using Majority Voting and Stacking separately which form the result of ensemble technique. Figures 3 and 4 show the framework of second phase using Majority Voting and Stacking respectively. We designed a SR converter module to convert the dataset which is available in IOB2 model into other SR module.

3.1 Feature extraction

Features, the properties of tokens or words, are the keystones of ML algorithms. The following features were extracted for our system:-

1. Word length:- This is a numeric value that determines the length of the current token.

2. Context words:- These are the words surrounding the current word. The context window of size n means n words before the current word and n words after the current word, e.g. context window of size 3

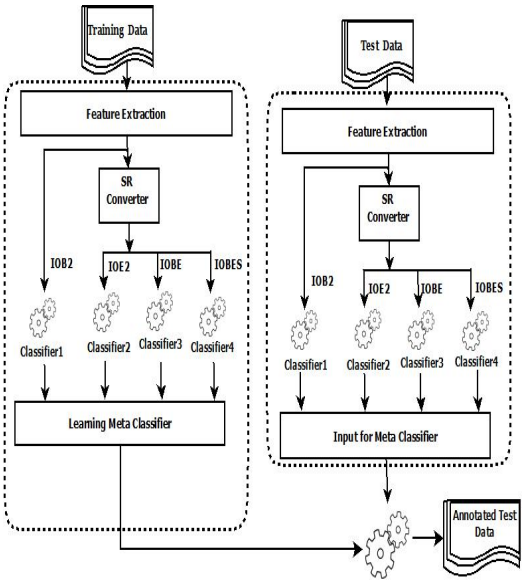


Figure 4: Combining base classifiers using Stacking

is $w_{i-3}...w_i...w_{i+3}$ where w_i is the current word.

3. Word affixes:- These are prefixes and suffixes of the current word. Prefix of length n is the first n characters of the word, while suffix of length n is the last n characters of the word. We have used all suffixes and prefixes up to length 5.

4. Part-of-Speech (PoS) tags:- PoS information is a very important feature, it determines the role of the word in the sentence. PoS tags are extracted using GENIA tagger V3.0.21[1].

5. Chunk Information:- Chunk information is useful when determining the boundaries of NEs. chunk information is extracted using GENIA tagger V3.0.21.

6. Word Normalization:- Two types of normalization namely stemming feature and word shape feature are used. Word stem means the root of a word. GENIA tagger V3.0.21 is used to extract the stems. There are two types of word shapes, general word shape and summarized word shape. In a general word X is substituted for each capital letter, x for each small character and d for consecutive digits. In a summarized word shape, consecutive capital letters are replaced by X, consecutive small letters by x and consecutive digits by d.

7. Orthographic features:- These features capture word formation information. The set of all orthographic features extracted are shown in Table 2.

8. Dynamic Feature:- It denotes the predicted tags of the words preceding the current word. This feature is calculated during running. An example of dynamic feature of size 4 are the tags $t_{-4}, t_{-3}, t_{-2}, t_{-1}$ corresponding to the words $w_{-4}, w_{-3}, w_{-2}, w_{-1}$, where current word is w_0.

9. Stop Words:- This is a logical feature which fires only if the current word is a stop word.

10. Non-Word:- This is a binary value which fires only if the word exists in entire dictionary. We used Grady augmented dictionary in `qdapDictionaries` package in `R` software (R Core Team, 2017).

11. Head Nouns:- The noun phrase describes the functionality or property of a clinical NE called head noun (Ekbal and Saha, 2013). For example, examination is the head noun of "*cardiovascular examination*". Head nouns are very important as these play a key role for correct classification of a clinical NE class. Unigrams and bigrams are used as head nouns. For domain dependency, training data is used to extract head nouns.

3.2 Support Vector machines

SVM is a binary classifier, which creates a hyperplane that discriminates between the two classes. SVM can be extended to multi-classes problems by combing several binary SVMs and combining using a one-vs-rest method or one-vs-one method (Hsu and Lin, 2002).

3.3 Evaluation Metrics

The performance of our system is reported in terms of f-measure (Hripcsak and Rothschild, 2005). F-measure is a harmonic mean of Precision (**P**) and Recall (**R**). Denoting **TP** as the number of true positives, **FP** number of false positives and **FN** as the number of false negatives, recall, precision and f-measure are calculated as follow:

$$P = \frac{TP}{TP + TF}$$

[1]http://www.nactem.ac.uk/GENIA/tagger/

Feature	Example
INITCAPS	Tonsillectomy
ALLCAPS	MCV, RBC
ENDCAPS	pH, proBNP
INCAPS	freeCa
CAPSMIX	cTropnT
HASDIGIT	pO2,calHCO3
HASHYPHEN	hyper-CVAD
ALPHNUM	B12
GREEK	alpha
NUMBER	101.5
HASATGC	LACTATE
PUNCT	INR(PT)
ROMAN	IV, CD

Table 2: List of orthographic features and examples

$$R = \frac{TP}{TP + FN}$$

$$f - measure = \frac{2 * P * R}{P + R}$$

3.4 Dataset

Our model is eval on i2b2 dataset (Uzuner et al., 2011), which was originally created for entity and relation extraction purposes at i2b2/VA 2010 challenge. It includes 826 discharge summaries for real patients from the University of Pittsburgh Medical Centre, Partners Health Care and Beth Israel Deaconess Medical Centre. Pittsburgh discharge summaries was used as a test set in i2b2 challenge and other two sources used as a training set. Statistics of the dataset is shown in Table 3. Both testing and training sets are manually annotated with three different named entities namely, treatment, problem and test. It is important to note that, there is lack of data sets that used for Clinical-NER.

4 Experiments and results

The proposed method combines the outputs of base-classifiers using two different approaches namely Majority Voting and Stacked Generalization.

For training the base classifiers, YamCha[2] toolkit along with TinySVM-0.092[3] is used. While training, a context window of size 3 is used (i.e. $w_{i-3}, w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, w_{i+3}$, where w_i is the current word) and the dynamic features are set at three (i.e. the output tags $t_{i-3}, t_{i-2}, t_{i-1}$ of the three words w_{i-3}, w_{i-2}, w_{i-1} preceding the current word w_i will be considered).

In Majority Voting, the out of all base classifiers are combined together and the output of final system is decided based on majority voting. If majority voting fail then the highest performance output of the base classifiers is considered on final output. For Stacked Generalization, an open source implementation of CRF, CRF++ package[4], has been used for constructing a CRF-based meta classifier.

The results of base classifiers and ensemble classifiers using Majority Voting and Stacking are shown in Table 4. The results show that, the best base classifier is the classifier based on IOBE SR model and the worst is the classifier based on IOE2 SR model. Both ensemble classifiers outperform the base classifiers and ensemble using stacking approach reported the best f-score.

5 Conclusion

Clinical-NER is a key task in health information systems. Different approaches have been applied for Clinical-NER. Ensemble approach tries to overcome the weakness of one approach by the strength of another. In our paper, we have designed an ensemble approach using majority voting and stacking techniques to combine the results of base classifiers. We have used SVM for learning base classifiers using different SR models and CRF classifier for learning the meta-classifier. Up to our knowledge, it is the first work that uses SR models for learning the base classifiers. The performance of our approach outperforms the performance of each of base classifiers.

References

Alan R Aronson and François-Michel Lang. 2010. An overview of metamap: historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229–236.

Frédéric Béchet, Alexis Nasr, and Franck Genet. 2000. Tagging unknown proper names using decision trees. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 77–84, Stroudsburg, PA, USA. Association for Computational Linguistics.

[2]http://chasen.org/taku/software/yamcha/

[3]http://chasen.org/taku/software/TinySVM/

[4]https://taku910.github.io/crfpp/

		Training set	Test set	Total
No. of Documents		349	477	826
Named Entities	Problem	11968	18500	30468
	Treatment	8500	13560	22060
	Test	7369	12899	20268

Table 3: Statistics of i2b2 dataset

Classifiers	SR Model	F-score
Base Classifier	IOB2	77.31
	IOE2	76.06
	IOBE	77.48
	IOBES	77.21
Ensemble Classifiers	Stacking	**77.63**
	Majority Voting	**77.53**

Table 4: Results of base and ensemble classifiers

Han-Cheol Cho, Naoaki Okazaki, Makoto Miwa, and Junichi Tsujii. 2013. Named entity recognition with multiple segment representations. *Information Processing & Management*, 49(4):954 – 965.

A. Dehghan, J. A. Keane, and G. Nenadic. 2013. Challenges in clinical named entity recognition for decision support. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 947–951, Oct.

Asif Ekbal and Sriparna Saha. 2013. Stacked ensemble coupled with feature selection for biomedical entity extraction. *Knowledge-Based Systems*, 46(0):22 – 32.

Carol Friedman, Philip O Alderson, John HM Austin, James J Cimino, and Stephen B Johnson. 1994. A general natural-language text processor for clinical radiology. *Journal of the American Medical Informatics Association*, 1(2):161–174.

R. Grishman and B. Sundheim. 1996. Message Understanding Conference-6: A Brief History. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, June.

Gurulingappa H, Hofmann-Apitius M, and Fluck J. 2010. Concept identification and assertion classification in patient health records. In *Proceedings of the 2010 i2b2/VA Workshop on Challenges in Natural Language Processing for Clinical Data*.

Scott Halgrim, Fei Xia, Imre Solti, Eithon Cadag, and Özlem Uzuner. 2010. Extracting medication information from discharge summaries. In *Proceedings of the NAACL HLT 2010 Second Louhi Workshop on Text and Data Mining of Health Documents*, Louhi '10, pages 61–67, Stroudsburg, PA, USA. Association for Computational Linguistics.

George Hripcsak and Adam S. Rothschild. 2005. Agreement, the f-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298.

Chih-Wei Hsu and Chih-Jen Lin. 2002. A comparison of methods for multiclass support vector machines. *Trans. Neur. Netw.*, 13(2):415–425, March.

S. Keretna, C. P. Lim, and D. Creighton. 2014. A hybrid model for named entity recognition using unstructured medical text. In *2014 9th International Conference on System of Systems Engineering (SOSE)*, pages 85–90, June.

Sara Keretna, Chee Peng Lim, Doug Creighton, and Khaled Bashir Shaban. 2015. Enhancing medical named entity recognition with an extended segment representation technique. *Computer Methods and Programs in Biomedicine*, 119(2):88 – 100.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

Dingcheng Li, Karin Kipper-Schuler, and Guergana Savova. 2008. Conditional random fields and support vector machines for disorder named entity recognition in clinical texts. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, BioNLP '08, pages 94–95, Stroudsburg, PA, USA. Association for Computational Linguistics.

Serguei Pakhomov, Ted Pedersen, and Christopher G Chute. 2005. Abbreviation and acronym disambiguation in clinical discourse. In *AMIA Annual Symposium Proceedings*, volume 2005, page 589. American Medical Informatics Association.

R. Polikar. 2006. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, Third.

R Core Team, 2017. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. *In proceeding of the Third ACL Workshop on Very Large Corpora*.

Adwait Ratnaparkhi. 1998. *Maximum entropy models for natural language ambiguity resolution*. Ph.D. thesis, University of Pennsylvania, PA, USA.

Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.

H. L. Shashirekha and H. A. Nayel. 2016. A comparative study of segment representation for biomedical named entity recognition. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1046–1052, Sept.

Stephan Spat, Bruno Cadonna, Ivo Rakovac, Christian Gütl, Hubert Leitner, Günther Stark, and Peter Beck. 2008. Enhanced information retrieval from narrative german-language clinical text documents using automated document classification. *Studies in health technology and informatics*, 136:473.

Jiashen Sun, Tianmin Wang, Li Li, and Xing Wu. 2010. Person name disambiguation based on topic model. In *CIPS-SIGHAN Joint Conference on Chinese Language Processing*, page 391.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556.

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.

Yonghui Wu, Min Jiang, Jianbo Lei, and Hua Xu. 2015. Named entity recognition in chinese clinical text using deep neural network. *Studies in health technology and informatics*, 216:624.

Yu-Chieh Wu. 2014. A top-down information theoretic word clustering algorithm for phrase recognition. *Information Sciences*, 275:213 – 225.

Shaodian Zhang and Noémie Elhadad. 2013. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of Biomedical Informatics*, 46(6):1088 – 1098.

Beyond Word2Vec: Embedding Words and Phrases in Same Vector Space

Vijay Prakash Dwivedi
CSED, MNNIT Allahabad
Allahabad, UP 211004, India
mail@vijaydwivedi.com.np

Manish Shrivastava
LTRC, IIIT Hyderabad
Hyderabad, TS 500032, India
m.shrivastava@iiit.ac.in

Abstract

Word embeddings are being used for several linguistic problems and NLP tasks. Improvements in solutions to such problems are great because of the recent breakthroughs in vector representation of words and research in vector space models. However, vector embeddings of phrases keeping semantics intact with words has been challenging. We propose a novel methodology using Siamese deep neural networks to embed multi-word units and fine-tune the current state-of-the-art word embeddings keeping both in the same vector space. We show several semantic relations between words and phrases using the embeddings generated by our system and evaluate that the similarity of words and their corresponding paraphrases are maximized using the modified embeddings.

1 Introduction

Vector embeddings in computational linguistics is a model that encodes words in a vector space. These vector encodings are used in mathematical models and serve as a base for computation in NLP.

Development of word embedding technique started in 2000 when Bengio et al. built neural probabilistic language models to reduce the high dimensionality of word representations in contexts by learning a distributed representation for words (Bengio et al., 2003). After that, continuous research has been done in the field resulting in remarkable improvements in word vector representations as well as the methods of learning the embeddings (Mikolov et al., 2013a; Mikolov et al., 2013b; Pennington et al., 2014). The primary reason for the increase in quality and performance of word vector embeddings is the huge growth of data and and development in computational capabilities as of today.

Natural language has both single word and multi-word units. If we want vector semantics to be near perfect, we need to embed multi-word units with the same quality as we do with the single word units. Improvements in phrase representation will eventually help the areas of question answering, search and translation. For a phrase that is similar to a certain word, the embedding of both the word and phrase should be similar and should lie in the same space. Only then a manipulation on a word and its paraphrase embedding can prove them to be similar.

Currently, compositional models are used to build phrase embeddings with less emphasis on building the compositions using deep learning and more using specific composition functions. Our major contribution in this work is employing deep neural architectures to transform constituent word embeddings of a multi-word units into its vector representation. We build a Siamese deep neural network architecture (Siamese LSTM, to be precise) that accepts two inputs, one being a word while another a phrase. The energy function in the Siamese network measures the similarity between these two input units. In the course of training the network, baseline word embeddings (Section 5.2) are modified and phrase embeddings are generated. We describe the model in detail in further sections.

2 Related Work

There has been a significant development in phrase embeddings after the word2vec breakthrough by Mikolov et al. in 2013. Earlier, word vectors were combined with some functions to create phrase vectors. (Mitchell and Lapata, 2008) developed systems with predefined composition operators. In their work, they created datasets of similarity for adjective-noun, noun-noun and verb-object bi-

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 205–211

gram units. They found the simple additive and multiplicative function to be quite effective. However, these simple functions ignored word orientation in phrases and their interaction.

To make these compositions robust in order to handle complex structures in sentences, Matrix composition functions (Zanzotto et al., 2010; Socher et al., 2012) and Tensor composition functions (Bride et al., 2015) were proposed. In 2013, Mikolov et al. generated phrase representation using the same method used for word representation in word2vec (Mikolov et al., 2013a; Mikolov et al., 2013b). High-frequency multi-word units such as *New York* was embedded along with the words by taking them as single token, or pseudo-words, i.e. *New_York*. Though this method is useful for learning short phrase representations with good quality, it does not generalize well to relatively longer and rare occurring phrases in the dataset.

In 2011, Socher et al. used a recursive neural network to learn representations for multi-word phrases. In particular, they used an unsupervised autoencoder and their model performed well on the sentiment classification task but not so well on phrase similarity related problems. The primary reason for this was the low-dimensional representations (upto 50) they had used to reduce the computational complexity. More recently (Yu and Dredze, 2015), the idea of learning composition functions based on phrase structure and context was proposed to compose phrase embeddings using baseline word embeddings.

The composition model developed by Yu and Dredze used Feature-rich Compositional Transformations (FCT) from words to phrases in which the summation weights were defined by the linguistic features of component words such as POS tags, head words and so on.

3 Methodology

We develop our model with the objective of training a system to predict similarity between a word and a phrase leveraging a similarity dataset. In our work, unlike the previous approaches, we capture the sequence of words in a phrase and their interaction as well. This is achieved by using a recurrent neural network to train our model. The model learns to generate phrase representations according to its closest single word meaning. The more the semantic similarity between the word and the phrase, the closer is their similarity metric output to 0 (1 otherwise). Table 1 shows some examples giving more insight into this.

Input word	Input phrase	Output
remorse	deep regret	0
athletes	bring up	1
suez	the suez canal	0
payment	earth orbit	1

Table 1: Input-Output samples. 'Output' is the similarity metric output, *i.e. 0 for similar and 1 for dissimilar*

We join the outputs of the two inputs fed to the network using a Siamese similarity function where the input of one sub-network is the baseline embedding of a word and the input of another subnetwork is the embeddings of constituent words of a phrase which are fed sequentially. The two outputs thus obtained are the resultant vector embeddings of the corresponding word and phrase generated by our system. The weights learned are common during both the inputs. In this way, we build a common abstract transformation for both the word and the phrase.

While training our system over a large dataset containing input-output pairs as in Table 1, the model learns weights with which we build phrase embeddings and fine-tune baseline word embeddings such that both are embedded in the same space.

3.1 Siamese Neural Network

For the task of signature verification, Siamese Neural Networks were first proposed by Bromley et al. in 1993. After that, the architecture has been used in several works of similarity and discrimation such as for face verification (Chopra et al., 2005), visual search (Liu et al., 2008), sentence similarity (Mueller and Thyagarajan, 2016), similar question retrieval in Q/A (Das et al., 2016), etc.

Suppose Out(X) is a set of functions which has a set of parameters *W*. In Figure 1, input A is first given to the network. Then another input B is fed and a similarity function gets the outputs of these A and B, i.e. Out(A) and Out(B). Siamese network learns a value of *W* such that the similarity metric is small if *Inp. A* (first input) and *Inp. B* (second input) are similar and large if they are not. The similarity function can be defined as:

$$S(Inp.A, Inp.B) = ||Out(A) - Out(B)|| \quad (1)$$

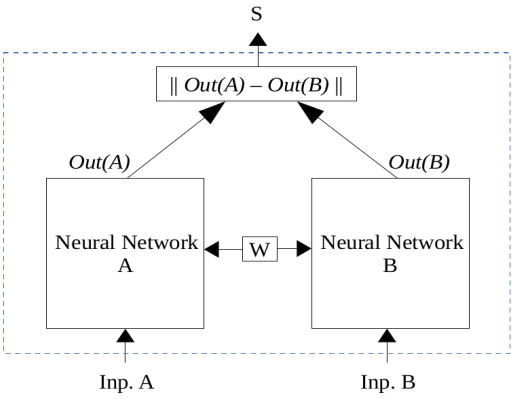


Figure 1: Siamese Neural Network

4 Model Architecture: Siamese LSTM

Long Short Term Memory Networks (LSTMs) (Hochreiter and Schmidhuber, 1997) are often used for problems with temporal (sequential) data. Since in our work, we are dealing with phrases which are sequences of words, we use LSTM network in our Siamese architecture. We first briefly describe LSTM networks and then the model architecture in detail.

4.1 LSTM Networks

Long Short Term Memory Network is a special variant of RNN which is capable of learning long-term dependencies. Each cell of LSTM contains four components: a memory state C_t, an output gate o_t that determines how the memory state affects further units, as well as an input gate i_t that controls what gets stored in and a forget gate f_t which determines what gets omitted from the memory based on each new input and the current state. Figure 2 shows the four components and their interaction with the inputs and past and future information.

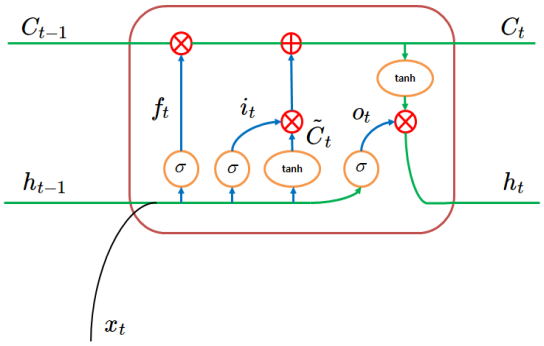


Figure 2: Block diagram of LSTM Cell

The use of LSTM network helps our system to learn the context of constituent words in a phrase.

4.2 Siamese LSTM

We use 3 layers of stacked LSTMs in our Siamese network. The hidden layers and the number of neurons were selected after repeated experiments and we obtained best results with this configuration. We limit the timesteps used in the LSTM to 5 since more than 99.8% of phrases in the dataset we use (PPDB[1]) are constituted of 5 or less units (words). Table 2 shows the composition of the PPDB data in terms of the n-grams in the phrases.

Size →	XL	
N-gram	**Nums**	**Percentage**
2-gram	2,98,536	79.40%
3-gram	60,657	16.13%
4-gram	13,132	3.49%
5-gram	2,993	0.80%
6-gram	682	0.18%
Total	3,76,000	100.0%

Table 2: Composition of the PPDB dataset of size XL showing only 0.18% of the phrases are of more than 5-grams; *the scores are 0.15% & 0.12% for XXL & XXXL sizes respectively*

At each timestep $t \in \{1 \ldots 5\}$, we use baseline embeddings of the word at t. For cases (*eg.* phrases of length less than 5 or word of length 1) where there is no word at t, we use zero embedding vector at that t.

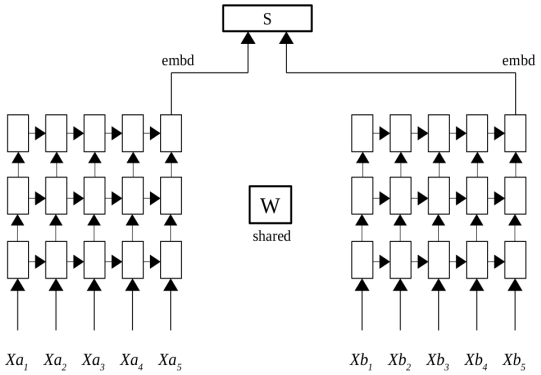


Figure 3: Siamese LSTM model architecture; First input to the network is a single word unit and second input is a multi-word unit

Figure 3 illustrates the model architecture we use. After training over a large dataset, the model

[1] http://www.cis.upenn.edu/~ccb/ppdb/

learns the set of parameters W. 'embd' is the resultant embedding of the word or paraphrase which is input to the network. The outputs of the two inputs to the network are joined using a contrastive loss function (Hadsell et al., 2006) which is defined as:

$$L = (1-Y)\frac{1}{2}(S)^2 + (Y)\frac{1}{2}\{max(0, m-S)\}^2 \quad (2)$$

where Y is a binary label assigned to a pair X_a and X_b (0 for similar and 1 for dissimilar), S is the similarity energy function which is parameterized by W and m is margin (Hadsell et al., 2006). We use $m = 1$. When two pairs are not similar the maximum energy metric outputs 1.

The gradient of the loss function with respect to W shared by the LSTM networks, is computed using back-propagation. Adamax optimizer, a variant of the Adam optimizer (Kingma and Lei Ba, 2015) is used to update the parameters of the sub-network.

5 Implementation

5.1 Dataset

We use PPDB dataset (Ganitkevitch et al., 2013) of size XL which has 3,76,000 pairs of words and their corresponding paraphrases. Since these are word-paraphrase pairs, we label the output of these pairs as 0. We augment the data by the same number of negative pairs by choosing a phrase for a word which is not its paraphrase. We label these pairs as 1. Thus, we train our model on 7,52,000 data samples.

5.2 Base Input Embedding

As base word embeddings for the input layers, we use GloVe[2] word vector embeddings (Pennington et al., 2014) of dimension 200.

The input pairs are passed through the Siamese network with the final layers of each network giving the resultant embeddings for words and phrases. We have 300 stacked neurons in the final layer of both the LSTM networks. This is the dimension of our resultant embeddings.

6 Experiments and Results

We perform the following experiments and find impressive results on various tasks.

[2] https://nlp.stanford.edu/projects/glove/

6.1 Word - Phrase Similarity Task

In this experiment, we define a classification task to determine if a word-phrase pair are semantically similar. We feed a word-phrase pair to the trained Siamese model and predict whether they are similar ($S \approx 0$) or dissimilar ($S \approx 1$).

We evaluate on the set of PPDB data (Section 5.1) left aside for validation. Out of the total data size, we choose 2,00,000 word-paraphrase pairs arbitrarily for the evaluation. Besides Siamese LSTM, we also perform this experiment on a Siamese Multi-layer Perceptron Network (MLP) where the MLP has 4 layers of neurons. As per our study in Table 2, we fix the MLP input to 5 words where each input word is in the form of a 200-D vector (we use padding and truncation for word units which are not of length 5). Therefore, the first layer has 1000 neurons. The remaining layers have 512, 512 and 300 neurons in order from second to final layer. We carry out the similarity task using Siamese MLP. However, we get better results (Figure 4) on Siamese LSTM which is also one reason why we chose it over Siamese MLP. We report the best accuracy of 76.65% on this task.

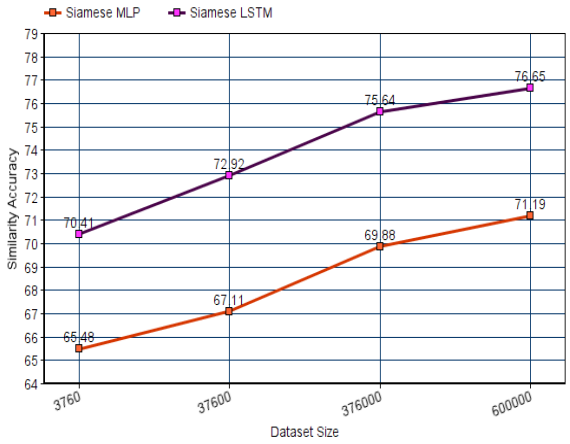


Figure 4: Accuracy of similarity over dataset size

6.2 Nearest Words and Phrases

In our work, we fine-tune the current base word embeddings while we generate phrase embeddings. Therefore, to validate the new vector embeddings of the words *(300-D)* which we obtain at the final layer of the Siamese sub-network, we perform this experiment.

For a pair $\langle U, V \rangle$, where U & V can be a single word or a multi-word unit, if U is given, we predict V (Refer Table 5). We output a list of four units which are closest to U in the vector space. We

Word	Nearest Words
viewpoints	perspectives, opinions, viewpoint, views
upbeat	optimistic, cautious, outlook, gloomy
sales	retail, selling, profits, profit
milder	colder, mild, warmer, heavier
1600	1400, 1700, 1300, 1500
asem	apec, asean, g20, summit
panelists	attendees, moderators, jurists, paragraphs
medal	medals, awarded, won, silver

Table 3: Nearest words for a given word showing that semantic relations are preserved even after the modification in the base embeddings.

Word	Nearest Phrases
viewpoints	differing views, the viewpoints, different opinions, different perspectives
upbeat	optimistic about, overly optimistic, more cautious, cautious
sales	sales volume, sales orders, export sales, the sales
milder	relatively mild, cold weather, even heavier, very mild
1600	1600 hours, 1300 hours, 1700 hours, 1100 hours
asem	the asem process, apec leaders, apec economies, summit meetings
panelists	discussion forums, two paragraphs, selected topics, panel discussion
medal	the gold medal, a gold medal, a medal, the bronze

Table 4: Nearest phrases for a given word: we show these for the same words as in Table 3 for the ease of comparison

perform this task by calculating the cosine similarity between the given U's embedding and all the units' embeddings in our vocabulary.

We finally select the top four ranked results for every U using the embeddings our model has generated and show interesting sample results in Table 3, Table 4 and Table 6. We notice several instances where semantics are preserved even after fine-tuning the baseline embeddings using our approach. The phrase representations generated from this work also stays close to its corresponding similar word's embedding.

Experiment	U	V
Table 3	Word	Word
Table 4	Word	Phrase
Table 6	Phrase	Phrase

Table 5: Experiments in this category; *word* is *1-gram* and *phrase* is *n-gram* where $n \geq 2$

Phrase	Nearest Phrases
are crucial	are important, are needed, are essential, are necessary
2005 to 2006	2005 to 2007, 2005 to 2008, 2003 to 2006, 2004 to 2005
the violence	the acts of violence, the violent, the prevalence of violence, the cycle of violence
both leaders	the two leaders, both members, the leaders, leaders of the two countries
president hosni mubarak	president mubarak, egyptian president hosni mubarak, hosni mubarak, the egyptian president hosni mubarak
the correctness	the objectivity, the veracity, the originality, the propriety
musical works	artistic works, musical instruments, works of art, works well

Table 6: Nearest phrases for a given phrase

6.3 Semantic Similarity Task

We use the embeddings derived by our system to evaluate the phrasal semantic similarity task of SemEval2013[3] and compare our results with that of the existing systems. The task of SemEval2013 5(a) is to determine if a word-phrase pair are semantically similar (True for similar and False for dissimilar) which is notably as same as the experiment in Section 6.1. We report the results (accuracy scores) of our system along with existing benchmark methods in Table 7. RAE is the recursive auto-encoder model developed in (Socher et al., 2011) wheras FCT-LM and FCT-Joint are the Feature Rich Compositional Transformation Models proposed by (Yu and Dredze,

[3] https://www.cs.york.ac.uk/semeval-2013

2015) with Language Modeling and Joint Traning (Language Modeling and Task-Specific) objective respectively for updating the embeddings. We also report our results with the Recursive Neural Network (ReNN) based model developed by (Socher et al., 2011; Socher et al., 2013) and obtain comparable results with the state-of-the-art system developed for generating phrase representations evaluated on this task.

Model	SemEval2013 Test
RAE	51.75
FCT-LM	67.22
FCT-Joint	70.64
ReNN	**72.22**
Our System	**72.14**

Table 7: Performance on the SemEval2013 5(a) Semantic Similarity Task

We see that the Siamese network based model outperforms the RAE by significant margin. However, the ReNN still has the best performance. Since the method proposed in this work is primarily dependant on the dataset containing word-paraphrase pairs, the larger this data size, the better quality embeddings we can generate and the performance on this task can be ultimately improved.

7 Conclusion and Discussion

In this work, we present a novel approach in building phrase vector embeddings by the use of its constituent word vectors through a sequential Siamese model. The Siamese network designed for this task leverages a word-phrase similarity dataset (PPDB) and generates embeddings of the phrase keeping in consideration the word position in the phrase, and its orientation and interaction with neighbour words as well which, in particular, is achieved using a Long Short Term Memory Network. Unlike previous attempts in building compositional models for phrase representation, the system presented in this paper does not employ any manual feature based technique for building phrase embedding or rely on a POS tag of particular word's neighbour or head words, or any other linguistic feature. Rather, we develop a similarity based deep learning network with contrastive loss which learns its weights after training and this set of learned parameters function as an abstract transformation which compose the phrase representation eventually. In addition, we fine-tune the base word vectors using the same abstract transformation and embed both the words and phrases in the same vector space.

The phrase representations derived from our model are computationally efficient as compared to recursive neural networks employed for this task. Besides, we are able to generate comparatively higher dimension (300-D) vectors in this work as compared to recursive networks (25-30D) which have a higher computational complexity. We show excellent results on phrase similarity task using the vector embeddings produced from this work. Also, semantic relationship between nearby words-phrases and phrases-phrases has been preserved.

8 Future Work

Since the network used in this work is trained on the paraphrase dataset, quality of phrase embeddings will improve if we use more exhaustive set and large number of word-paraphrase pairs for training. If we are able to extract more paraphrases using bigger language corpus and use it for training our system, we can considerably improve the quality of vectors derived. In future, we plan to extend this work to other languages as well by first extracting paraphrases and then learning a similarity model to derive phrase representations. However, efficient and reliable word vectors of higher dimension is required for this work to be done. Similarly, we intend to use phrase embeddings generated by our model in several NLP applications with motive of improving the performance.

Acknowledgement

We would like to thank Naveen Kumar Laskari for discussions during the course of this work and Pruthwik Mishra and Saurav Jha for their valuable suggestions.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research (JMLR), 3:1137-1155.*

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546.*

Jeffrey Pennington, Richard Socher and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Association for Computational Linguistics (ACL)*, pages 236-244.

Mo Yu and Mark Dredze. 2015. Learning Composition Models for Phrase Embeddings. In *Transactions of the Association for Computational Linguistics (ACL)*, pages 227-242.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Sackinger, and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4).

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:539–546.

Yuanjie Liu, Shasha Li, Yunbo Cao, Chin-Yew Lin, Dingyi Han, and Yong Yu. 2008. Understanding and summarizing answers in community-based question answering services. In *International Conference on Computational Linguistics (COLING)*, 1:497–504.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In *AAAI Conference on Artificial Intelligence*.

Arpita Das, Harish Yenala, Manoj Chinnakotla and Manish Shrivastava. 2016. Together We Stand: Siamese Networks for Similar Question Retrieval. In *Association for Computational Linguistics (ACL)*, pages 378-387.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Juri Ganitkevitch, Benjamin Van Durme and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT*, pages 758–764.

Raia Hadsell, Sumit Chopra, Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2:1735–1742.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 1201–1211.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Ng. 2013. Parsing with compositional vector grammars. In *In Association for Computational Linguistics (ACL)*, pages 455–465.

Antoine Bride, Tim Van de Cruys, and Nicholas Asher. 2015. A Generalisation of Lexical Functions for Composition in Distributional Semantics. In *Association for Computational Linguistics (ACL)*, pages 281–291.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. In *The International Conference on Learning Representations (ICLR)*.

Fabio M. Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating Linear Models for Compositional Distributional Semantics. In *International Conference on Computational Linguistics (COLING)*, 1263-1271.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Empirical Methods in Natural Language Processing (EMNLP)*, 151-161.

Relationship Extraction based on Category of Medical Concepts from Lexical Contexts

Anupam Mondal **Dipankar Das** **Sivaji Bandyopadhyay**
Department of Computer Science and Engineering
Jadavpur University, Kolkata, India
link.anupam@gmail.com, dipankar.dipnil2005@gmail.com,
sivaji_cse_ju@yahoo.com

Abstract

Medical information extraction is an emerging task in healthcare services aim to acquire crucial information of the concepts like *diseases, symptoms, and drugs* and also to identify their *relations* from corpora. In the present article, we have proposed a relationship extraction system based on such categories of medical concepts. We have employed rule-based as well as Support Vector Machine (SVM) based feature-oriented approach along with a domain-specific lexicon viz WordNet of Medical Event (WME 2.0). The lexicon assists in recognizing medical concepts and their related features like Parts-Of-Speech (POS), categories, and Similar Sentiment Words (SSW). We have opted only four fundamental categories *diseases, drugs, symptoms, and human_anatomy* of medical concepts as provided in WME lexicon. Such categories play a crucial role in identifying eight types of different semantic relations viz. *drug-drug, disease-drug, and human_anatomy-symptom* from the medical context. Thereafter, we have validated both rules and features-oriented approaches and offers an average F-Measures of 0.79 and 0.86 individually.

1 Introduction

The availability of medical documents such as reports, discharge summaries, and prescriptions and their related information are growing quickly. In order to extract critical and crucial information, the researchers have applied various statistical and ontology-based approaches with well-known machine learning classifiers (Mondal et al., 2016b; Uzuner et al., 2011). The extracted informations are medical concepts (terms), categories (classes), and their relations, which assist the experts such as doctors and other medical practitioners as well as the non-experts as patients in understanding the problems (e.g. *diseases, symptoms*) and their related remedies (e.g. *drugs*).

The medical concepts are presented by the key terms like words or phrases of the corpus whereas the category refers to the fundamental classes of medical concepts such as diseases and symptoms. The assigned categories of medical concepts and their in-between relations help to build a medical annotation system. Besides, each sentence of the corpus is presented as a medical context in this paper. For an example, "***abdominal_pain***" denotes the medical concept and its category is denoted by "*symptom*" in the following medical context.

"***Abdominal_pain*** *is a* ***sign*** *of early* ***pregnancy***.".

In order to design our category based relationship extraction system, we observed the following major challenges:

A. The first challenge was how to identify the medical concepts and their textual spans from unstructured or semi-structured medical corpora. To address this challenge, we have used WordNet of Medical Events (WME), a domain-specific lexicon (Mondal et al., 2016a, 2015). The lexicon provides a good coverage while extracting medical concepts from our experimental dataset viz. SemEval-2015 Task-6[1] and MedicineNet[2].

B. The second challenge was how to decide the set of categories for the medical concepts and assign them. To overcome the first sub-challenge, we adopted the help of a group of medical practitioners and to cope-up with the second sub-

[1] http://alt.qcri.org/semeval2015/task6/
[2] http://www.medicinenet.com/script/main/hp.asp

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 212–219

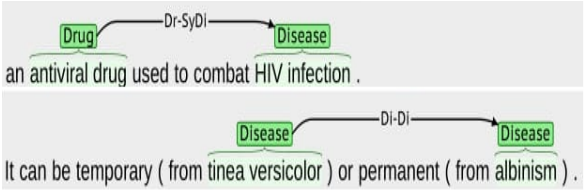


Figure 1: An example of extracted relations using proposed system.

challenge, we used the categorization system of medical concepts (Mondal et al., 2017, 2016a). The categorization system assists in assigning one of the four medical categories such as *diseases, symptoms, drugs, and human_anatomy* to the concepts.

C. The third and final challenge of the present work was how to identify the relations between a pair of medical concepts in a context and evaluate the relations. To address this issue, we have built a rule-based and a feature-based relationship extraction systems, which help to predict the type of relations between a pair of medical concepts. Table 1 shows the proposed eight types of relations with illustrative examples. Besides, we have manually prepared a labeled dataset, which contains 2000 medical contexts and their tagged medical concepts, categories, and relations as shown in Figure 1.

In the present research, our primary motivation was to build an annotation system which helps to assign all four types of medical categories such as *diseases, symptoms, drugs, and human_anatomy* and their related relations in a context. According to the best of our knowledge, we are unable to find any medical corpora which contain all the mentioned categories at a time. Afterwards, we have discussed the contribution of the paper as follows,

I. A labeled dataset preparation by a group of medical practitioners. The dataset has been labeled with medical concepts and their categories and proposed eight types of category-based relations in a context. We have acquired the dataset from SemEval-2015 Task-6 and MedicineNet resources which contain around 2000 number of medical contexts. The dataset helps to design and validate the relationship extraction system.

II. Relationship extraction plays a key role in identifying the semantic information from the corpus. To extract these relations, we have proposed a linguistic rule-based (Abacha and Zweigenbaum, 2011a; Hearst, 1992) and a feature-oriented machine learning (Rink et al., 2011; Zhu et al., 2009) approach. The rule-based patterns help to identify the specific relations from the dataset, whereas machine learning approach assists in extracting generalize relations with promising accuracy. For an example, the following medical context is able to extract *disease - symptom* (illustration, inflammation symptom for the adnexitis disease) and *symptom - human_anatomy* (illustration, inflammation affect the uterus) relations.

"*The* ***adnexitis*** *_disease characterizes* ***inflammation*** *_symptom of attachments of the* ***uterus*** *_human_anatomy.*"

The proposed relation extraction system assists in understanding the subjective information of the corpus. Besides, these systems guide to build various applications namely, annotation and recommendation system in healthcare services.

The overall structure of the paper is as follows. Section 2 presents the related work carried out in this field. Section 3 describes the dataset preparation and brat representation technique. Section 4 and Section 5 present the proposed relation extraction system and its evaluation approach. Finally, Section 6 presents the concluding remarks related to our study.

2 Related Work

2.1 Medical Ontologies and Lexicons

Biomedical information extraction research is challenging due to the availability of a large number of daily produced unstructured and semi-structured medical corpus. To represent the structured corpus and extracting the subjective and conceptual information from the corpus, a domain-specific lexicon is essential (Borthwick et al., 1998). To this end, the standard vocabularies and ontologies such as UMLS (Unified Medical Language System) and SNOMED-CT (Systematized Nomenclature of Medicine-Clinical Terms), and lexicons like MEN (Medical WordNet) and WME (WordNet of Medical Event) have used by the researchers (Smith and Fellbaum, 2004; Kilgarriff and Fellbaum, 2000; Mondal et al., 2016a).

2.2 Medical Category and Relation Extraction

These ontologies and lexicons assist in extracting the relevant information from the corpus such as medical concept categories and relations between medical concepts.

Relation	Explanation and Example
Dr-SyDi	A drug how helps to improve or cure or side effects the diseases or symptoms.
	Warfarin *is also used to reduce the risk of* ***clots*** *causing* ***strokes*** *or* ***heart_attacks****.*
Ha-SyDi	A disease or symptom which effects a part of the body.
	A ***painful_inflammation*** *of the* ***big_toe*** *and* ***foot****.*
Di-Sy	The symptoms which reflect a disease.
	Anal_fissures *typically cause* ***pain*** *and* ***bleeding*** *with* ***bowel_movements****.*
Dr-Dr	How the drugs are related each other.
	An ***oral_lipid-lowering_medicine*** *(* ***trade_name_Zocor*** *) administered to reduce blood cholesterol levels; recommended after heart attacks.*
Sy-Sy	How the symptoms are related each other.
	A ***rhythmic_tightening*** *in* ***labor_of_the_upper_uterine_musculature*** *that contracts the size of the uterus and pushes the fetus toward the birth canal.*
Di-Di	How the diseases are related each other.
	Kaposi's_sarcoma *is a form of* ***skin_cancer*** *that can involve internal organs.*
MMT-SyDi	The medical terms such as process and chemical components etc. how helps to refer the diseases or symptoms.
	When you swallow or inhale these highly ***toxic_products****, you can experience* ***life-threatening_symptoms****.*
Ha-Ha	How various body parts are related or effected each other under a situation.
	Nodding movement of the ***head*** *or* ***body****.*
Note: Di− > Disease, Dr− > Drug, Sy− > Symptom, Ha− > Human_anatomy, and MMT− > Miscellaneous Medical Term	

Table 1: An illustration of the proposed eight types of relations for the medical context.

Eklund (Eklund, 2011) developed an annotation system to extract the relations as *diseases* for *treatments* from the scientific medical corpus. Yao, et al. (Yao et al., 2010) extracted category relations as *cures, prevents, and side effects*, which describe the distinctive nature for the biomedical text (medical papers) (Abacha and Zweigenbaum, 2011b; Frunza and Inkpen, 2010). Franzen et al. (Franzén et al., 2002) have annotated Yapex corpus with 200 medical abstracts to extract the category as *proteins*. These ontologies are fundamentally looking for extracting *protein-protein* interaction and *disease-treatment* relations from corpora under a BioText project (Rosario and Hearst, 2005).

Khoo et al. (Khoo et al., 2000) developed a causal relations extraction system from abstracts of biomedical articles by aligning manually constructed graph patterns with syntactic dependency trees. Lee et al. (Lee et al., 2003) applied UMLS to identify semantic relations between medical entities. Their first method was able to extract 68% of the semantic relations in their test corpus but if many relations were possible between the relation arguments no disambiguation performed. Their second method (Lee et al., 2004) targeted the precise extraction of "treatment" relations between drugs and diseases. Manually written linguistic patterns were constructed from medical abstracts talking about cancer. Their system reached 84% recall but an overall 48.14% precision. Embarek and Ferret (Embarek and Ferret, 2008) developed an approach to extract four kinds of relations (*Detect, Treat, Sign, and Cure*) between five kinds of medical entities. The employed patterns were constructed automatically using an alignment algorithm which maps sentence parts using an edit distance (defined between two sentences) and different word-level clues.

3 Dataset Preparation

The present section describes how we employed a domain-specific lexicon namely WME 2.0 and prepared an annotated dataset for relation extraction system. Also, discussed the brat environment to visualize the tagged concepts, categories and relations of our medical corpora.

Evaluation Data: We initially acquired corpora from SemEval-2015 Task-6 and MedicineNet resources. Thereafter, the corpora have been converted into medical contexts in the form of single sentences according to the presence of medical concepts for our experiment. Table 2 shows the distributions of medical concepts and contexts from SemEval, MedicineNet as well as WME 2.0 resources.

We randomly collected 2000 medical contexts from the acquired corpora and manually labeled

	SemEval-2015 Task-6	MedicineNet	WME 2.0
Medical concepts	9786	9834	10186
Contexts (medical + non-medical)	10985	9076	-
Medical contexts	6774	7042	-

Table 2: A statistical distribution of unique number of the medical concepts and contexts from various resources.

by a group of medical practitioners in the brat environment. Table 3 shows the distributions of manually labeled category-based relations. In order to label the corpus, the medical practitioners have used WME 2.0 lexicon, which assists in understanding medical concepts and their categories based on their glosses and semantics.

Relation	Manually labeled
All relation	2071
Dr-SyDi	52
Ha-SyDi	198
Di-Sy	312
Dr-Dr	15
Sy-Sy	132
Di-Di	282
MMT-SyDi	927
Ha-Ha	153

Table 3: A statistics of manually labeled various relations

WME Lexicon: In healthcare, a lexicon from the medical domain is demanding to identify the conceptual information such as category or sentiment from the corpus (Cambria, 2016). To this end, we borrow the knowledge from WordNet of Medical Event (WME 2.0), a domain-specific lexicon (Mondal et al., 2016a, 2015).

However, the current version of WME namely WME 2.0 was enhanced with more sentiment and semantic features for 10186 number of medical concepts (Mondal et al., 2017, 2016a). WME 2.0 was added with affinity score, gravity score, Similar Sentiment Words (SSW), and category feature along with the existing features of WME 1.0, e.g., gloss (descriptive explanation), Parts-Of-Speech (POS), polarity score, and sentiment.

The conventional WordNet [3], a preprocessed medical dictionary, and SenticNet [4] were applied to prepare our present resource for extracting semantic relations of concepts. Affinity score indicates what extend two concepts are close to each other by measuring the number of common sentiment words (SSW) appeared for a pair of concepts within the range of 0 to 1. On the other hand, gravity score identifies sentiment-oriented relevance between medical concepts and their various glosses (descriptive explanations) and ranges from -1 to 1. While -1 suggests no relation and 1 indicates strong relations either positive or negative, which helps to identify a proper gloss for concepts. Besides, the assigned categories such as *diseases, drugs, treatments, human_anatomy,* and *MMT* assist in extracting the subjective information of the concepts.

For example, WME 2.0 lexicon presents the properties of a concept say ***amnesia*** as of category (*disease*), POS (*noun*), gloss (*"loss of memory sometimes including the memory of personal identity due to brain injury, shock, fatigue, repression, or illness or sometimes induced by anesthesia."*), SSW (*memory_loss, blackout, fugue, stupor*), polarity score (*-0.375*), affinity score (*0.429*), gravity score (*0.170*), and sentiment (*negative*).

Brat Annotation: We have used an annotation tool namely brat to manually label the relations between a pair of medical concepts within their contexts. The tool helps to easily label the contexts, which generate an annotation (.ann) file for each input text (.txt) file. The .ann file has labeled medical concepts along with their IDs (Ti), categories (Disease, Drugs etc.), textual spans and concepts and relations with IDs, categories and arguments. Thereafter, we have written a python script to convert the manually tagged annotation file into the format according to our proposed system output. The script assists in evaluating the proposed extracted relations.

For example, the following medical context denotes an annotated output as

"T1 Disease 1 39 Giant cell interstitial pneumonia (GIP)"

"T2 Disease 59 77 pulmonary_fibrosis"

"R1 Di-Di Arg1:T1 Arg2:T2".

"***Giant_cell_interstitial_pneumonia_(GIP)*** *_disease is a rare form of* ***pulmonary_fibrosis*** *_disease.*"

4 Relationship Extraction

Biomedical texts are primarily rich with subjective information such as *problems and treatments* and they are represented as medical concepts, category and their relations in case of ours. Recognition of important relations between medical concepts is a challenging task due to lack of involvement of domain-experts. Thus, to overcome the challenge, we have proposed a relation extraction system by utilizing the categories of medical concepts. To

[3] https://wordnet.princeton.edu/

[4] http://sentic.net/

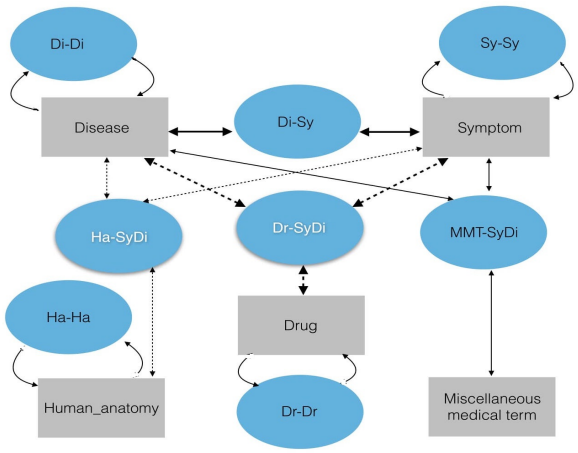


Figure 2: A flow diagram of different types of relations in medical context.

develop the category-based relation extraction system, we have considered the following hypothesis and proposed methodology viz. rule-based and feature-oriented approach.

4.1 Hypothesis:

In the present research, we have considered four primary categories of medical concepts *diseases, drugs, human_anatomy, and symptoms* and a combined category *MMT* that refers to unrecognized categories of medical concepts. To identify the relationship between these categories, we have adopted eight types of relations after close observations done by experts. Figure 2 shows the overall presentation i.e how we proposed eight relations based on various concepts and their categories in a context. Finally, we have classified these relations into two major groups as combined and direct. The combined relations are *Ha-SyDi, Dr-SyDi*, and *MMT-SyDi* and rest of the five relations are presented as direct relation as mentioned in Table 1.

We have also observed that two of the categories (e.g., Disease and Symptom) are very close to each other in their context level appearances and therefore we merged them to make a single category (e.g., SyDi) instead of making the individual pair of relation with other categories.

4.2 Proposed Methodology:

Rule-based Approach In case of relation extraction, rule-based approach adopts various linguistic textual patterns between the pair of medical concepts. To identify these patterns, we have collected the promising pairs of the medical category that are semantically or subjectively related each other as shown in Figure 2. The consecutive appearance of concepts has not been taken into account in case of identifying rules because such concepts are ambiguous in nature and conflicting in their medical senses.

These identified patterns converted to Static Surface Patterns (SSP) using various regular expressions and are labeled by our proposed relations. An example, the linguistic textual pattern "*<Drug> used to combat <Disease>*" is converted to static surface pattern as "*<Drug>(.*?)<Disease>*" with *Dr-SyDi* relation label.

The linguistic patterns help to reduce the manual effort and enhance the list of patterns with new relations where SSP assists in designing an automated relationship extraction system. Table 4 presents the number of extracted SSPs for eight relations with the specific example.

Therefore, the following algorithm has been applied to extract the category-based relations between the pair of medical concepts in a context. The output of the proposed system is shown in Figure 3.

Algorithm:

1. Identify the category of annotated medical concepts in a context and present them as C = {MC1,MC2,...,MCn}, where MCn is nth identified medical concept with category in a context.

2. Compare the Static Surface Patterns (SSP) with the pair of medical concepts from C.

2.1. If SSP matches with the pair of concepts in C, then assigned the corresponding relation.

2.2. Else look for the next pair of concepts in C as mentioned in Step 2.

3. If not found any relation in C then label "*No relation*" and move to next context (C).

4. Else combine the assigned relations with labeled concepts for the context (C) and move to next context (C).

Feature-oriented Approach Relation extraction is presented as a multi-label classification problem due to the presence of various types of semantic relations between medical concepts. To address this problem, we have considered feature-oriented machine learning approach over rule-based approach. The concerned features are category, intermediate word sequence, POS, and SSW of the pair of medical concepts in a context for our experiment.

Relation	Patterns	Example
Dr-SyDi	12	.... <*Warfarin/Drug*> *is also used to reduce the risk of* <*clots/Symptom*> ....
Ha-SyDi	14	.... <*coronary artery/Human_anatomy*> *by a* <*thrombus/Disease*> ....
Di-Sy	24	.... <*halitosis/Disease*> *, is an* <*unpleasant odor/Symptom*>....
Dr-Dr	6	.... <*oral lipid-lowering medicine/Drug*> *trade name* <*Zocor/Drug*> ....
Sy-Sy	10	.... <*Frozen shoulder/Symptom*> *, also known as* <*adhesive capsulitis/Symptom*> ....
Di-Di	18	.... <*Kaposi's sarcoma/Disease*> *is a form of* <*skin cancer/Disease*> ....
MMT-SyDi	27	.... <*toxic products/MMT*> *, you can experience* <*life-threatening symptoms/Symptom*> ....
Ha-Ha	6	.... <*vagus nerve/Human_anatomy*> *included , emerge from or enter the* <*skull/Human_anatomy*> ....

Table 4: A statistics and examples of relation patterns in the contexts.

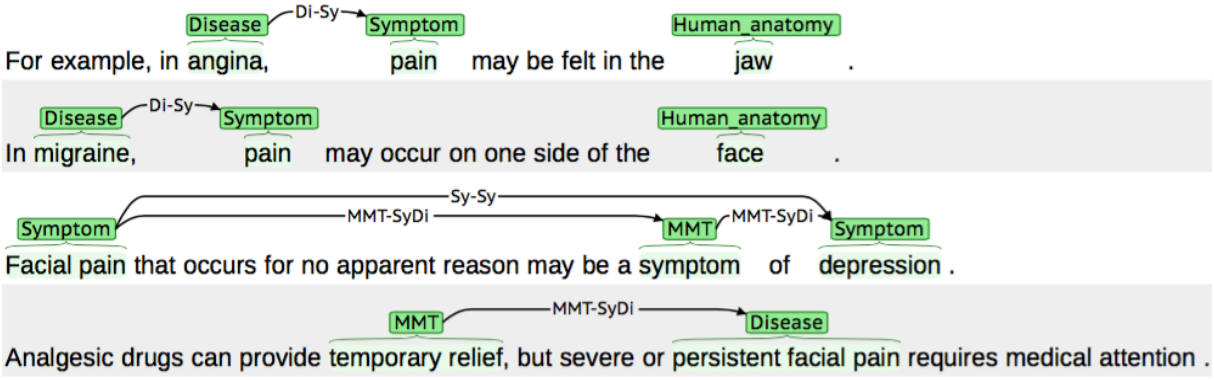


Figure 3: Output of the extracted relation using rule-based approach.

To identify the mentioned features, we have also employed WordNet and WME 2.0 lexicons. These lexicons help to assign the category, POS, and Similar Sentiment Words (SSW) for the medical concepts. Besides, we have written a python script to recognize the intermediate word sequence between the pair of the concepts.

For example, the following medical context identifies the features say 1. annotated medical concepts ("***degenerative_brain_disorder***" and "***dementia***") 2. POS labels (noun and verb) 3. intermediate word sequence ("(.*) that leads to (.*)") 4. categories of medical concepts (*disease* and *disease*), and 5. SSW ("***Alzheimer's disease***, ***Huntington's disease***, *and* ***Parkinson's disease***" and "***mental_illness***, ***madness***, *and* ***insanity***"), respectively.

"***Degenerative_brain_disorder*** *that leads to* ***dementia***."

Figure 4 illustrates the steps to extract the relations between a pair of medical concepts using proposed feature-oriented approach along with machine learning classifier. Besides, we have extracted these features from the evaluation dataset and processed through the linear Support Vector Machine (SVM) classifier to predict the relations. The following section discusses the evaluation process for both of the proposed approaches.

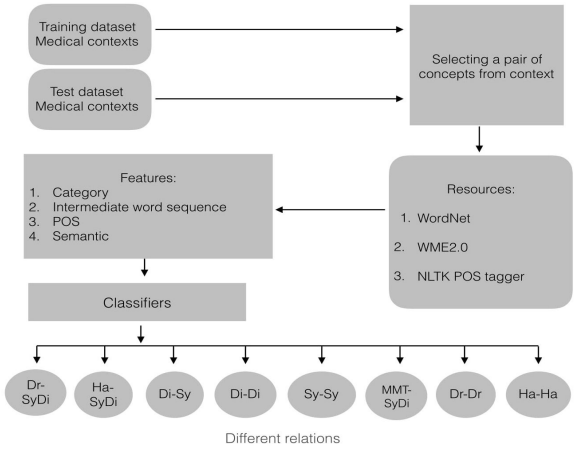


Figure 4: A flow diagram of the feature-oriented relationship extraction system.

5 Evaluation

We have used the state-of-the-art evaluation metrics such as precision, recall, and F-Measure [5] to validate the proposed relation extraction approaches.

Rule-based Relation Extraction In order to reduce the ambiguity of the extracted relations,

[5]https://en.wikipedia.org/wiki/Precision_and_recall

we have employed the same annotated medical concepts and their categories in both rule-based as well as feature-oriented approaches. Table 5 presents the distribution of manual and system tagged category-based relations and their related precision, recall, and F-measure.

Relation	Manually labeled	Extracted/Correct/Incorrect	Precision	Recall	F-Measure
All relations	2071	2681 / 1881 / 780	0.70	0.91	0.79
Dr-SyDi	52	102 / 46 / 56	0.45	0.88	0.59
Ha-SyDi	198	233 / 178 / 35	0.76	0.90	0.82
Di-Sy	312	386 / 282 / 104	0.73	0.90	0.81
Dr-Dr	15	22 / 13 / 9	0.59	0.87	0.70
Sy-Sy	132	193 / 115 / 78	0.60	0.87	0.71
Di-Di	282	341 / 254 / 87	0.74	0.90	0.81
MMT-SyDi	927	1227 / 871 / 356	0.71	0.94	0.81
Ha-Ha	153	177 / 122 / 55	0.69	0.80	0.74

Table 5: A statistics of various relation identification between the pair of medical concepts in context using rule-based approach.

Feature-oriented Relationship Extraction Besides, to validate the feature-oriented system, we additionally prepared a test dataset along with our built-in evaluation dataset. The test dataset contains rest of 11816 medical contexts as referred in Table 2.

Thereafter, the features have been extracted from the evaluation dataset and processed through linear Support Vector Machine (SVM) classifier to learn the proposed relation extraction model. Hence, the test dataset has been applied to the learned model for predicting and validating the extracted relations. Table 6 summarizes the result as precision, recall, and F-Measure.

Relation	Precision	Recall	F-Measure
All relations	0.92	0.81	0.86
Dr-SyDi	0.90	0.72	0.80
Ha-SyDi	0.91	0.80	0.85
Di-Sy	0.90	0.79	0.84
Dr-Dr	0.88	0.76	0.82
Sy-Sy	0.89	0.78	0.83
Di-Di	0.91	0.80	0.85
MMT-SyDi	0.93	0.82	0.87
Ha-Ha	0.88	0.72	0.79

Table 6: A statistics of various relation identification between the pair of medical concepts in context using feature-oriented approach.

Finally, we have observed that the feature-oriented approach provides a better F-Measure (0.86) over the rule-based approach (0.79) for extracting relations. So, we conclude that both approaches are important to extract the mentioned eight relations from the unstructured corpus.

6 Conclusion and Future Scope

In this article, we have focused on extracting eight types of category-based relations of medical concepts from the context. The relation extraction system facilitates to design various domain-specific applications. We have employed two well-known approaches such as rule-based and feature-oriented. Also, we have manually prepared an evaluation dataset to design and validate the relation extraction system. The rule-based approach helps to understand the semantic knowledge where linguistic features assist in improving the accuracy of the system. Finally, the evaluation section shows the effectiveness of the proposed relation extraction approaches by offering the average F-Measures of 0.79 and 0.86 for the rules and features-oriented techniques, respectively in healthcare. In future, we will try to introduce new relations for the medical concepts to build a relation database, which helps to design a medical recommendation system.

References

Asma Ben Abacha and Pierre Zweigenbaum. 2011a. Automatic extraction of semantic relations between medical entities: a rule based approach. *Journal of biomedical semantics* 2(5):1.

Asma Ben Abacha and Pierre Zweigenbaum. 2011b. A hybrid approach for the extraction of semantic relations from medline abstracts. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, pages 139–150.

Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proc. of the Sixth Workshop on Very Large Corpora*. volume 182.

Erik Cambria. 2016. Affective computing and sentiment analysis. *IEEE Intelligent Systems* 31(2):102–107.

Ann-Marie Eklund. 2011. Relational annotation of scientific medical corpora. In *LOUHI 2011 Third International Workshop on Health Document Text Mining and Information Analysis*. page 27.

Mehdi Embarek and Olivier Ferret. 2008. Learning patterns for building resources about semantic relations in the medical domain. In *LREC*.

Kristofer Franzén, Gunnar Eriksson, Fredrik Olsson, Lars Asker, Per Lidén, and Joakim Cöster. 2002. Protein names and how to find them. *International journal of medical informatics* 67(1):49–61.

Oana Frunza and Diana Inkpen. 2010. Extraction of disease-treatment semantic relations from biomedical sentences. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*. Association for Computational Linguistics, pages 91–98.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pages 539–545.

Christopher SG Khoo, Syin Chan, and Yun Niu. 2000. Extracting causal knowledge from a medical database using graphical patterns. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 336–343.

Adam Kilgarriff and Christiane Fellbaum. 2000. Wordnet: An electronic lexical database.

Chew-Hung Lee, Christopher Khoo, and Jin-Cheon Na. 2004. Automatic identification of treatment relations for medical ontology learning: An exploratory study. *ADVANCES IN KNOWLEDGE ORGANIZATION* 9:245–250.

Chew-Hung Lee, Jin-Cheon Na, and Christopher Khoo. 2003. Ontology learning for medical digital libraries. In *International Conference on Asian Digital Libraries*. Springer, pages 302–305.

Anupam Mondal, Erik Cambria, Dipankar Das, and Sivaji Bandyopadhyay. 2017. Auto-categorization of medical concepts and contexts. *Research in Computing Science* .

Anupam Mondal, Iti Chaturvedi, Dipankar Das, Rajiv Bajpai, and Sivaji Bandyopadhyay. 2015. Lexical resource for medical events: A polarity based approach. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, pages 1302–1309.

Anupam Mondal, Dipankar Das, Erik Cambria, and Sivaji Bandyopadhyay. 2016a. Wme: Sense, polarity and affinity based concept resource for medical events. *Proceedings of the Eighth Global WordNet Conference* pages 242–246.

Anupam Mondal, Ranjan Satapathy, Dipankar Das, and Sivaji Bandyopadhyay. 2016b. A hybrid approach based sentiment extraction from medical context. In *4th Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2016), IJCAI 2016 Workshop, July 10, Hilton, New York City, USA*.

Bryan Rink, Sanda Harabagiu, and Kirk Roberts. 2011. Automatic extraction of relations between medical concepts in clinical texts. *Journal of the American Medical Informatics Association* 18(5):594–600.

Barbara Rosario and Marti A Hearst. 2005. Multi-way relation classification: application to protein-protein interactions. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 732–739.

Barry Smith and Christiane Fellbaum. 2004. Medical wordnet: a new methodology for the construction and validation of information resources for consumer health. In *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, page 371.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association* 18(5):552–556.

Lin Yao, Cheng-Jie Sun, Xiao-Long Wang, and Xuan Wang. 2010. Relationship extraction from biomedical literature using maximum entropy based on rich features. In *2010 International Conference on Machine Learning and Cybernetics*. IEEE, volume 6, pages 3358–3361.

Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. 2009. Statsnowball: a statistical approach to extracting entity relationships. In *Proceedings of the 18th international conference on World wide web*. ACM, pages 101–110.

A Sinhala Word Joiner

Rajith Priyanga **Surangika Ranatunga** **Gihan Dias**
Department of Computer Science and Engineering
University of Moratuwa
Sri Lanka
rpriyanga@yahoo.com, surangika@cse.mrt.ac.lk, gihan@uom.lk

Abstract

Sinhala is an agglutinative language where many words are formed by joining several morphemes. Word joining is a basic operation in Sinhala morphology, and is based on sandhi rules.

The Sinhala word joiner is a software component which implements sandhi rules to synthesise a word from two or more morphemes. We studied Sinhala word join rules based on grammar and usage and implemented a library and a standalone application to synthesise Sinhala words. In addition to the joined word, it also outputs the rule used for joining. The tool uses a combination of a corpus and hand-coded rules to improve accuracy.

Keywords: Sinhala word joiner, Morphophonemic tools, corpus based scoring algorithm, sandhi rules

1 Introduction

Sinhala belongs to the Indo Aryan sub branch of the Indo-European language family. It is a descendent of the Sanskrit language, but was heavily influenced by the Pāli language from the second century B.C. as a result of the introduction of Buddhism to Sri Lanka. Other than from Pāli, Sinhala was influenced mainly by Tamil, Arabic, Portuguese, Dutch and English languages. Sinhala is written in its own script which is a descendent of the Brahmi script.

Even though the Sinhala language and its script have many similarities with their ancestors from India, they have evolved uniquely over two millennia.

There have been some attempts to implement morphological synthesizers and analysers for Sinhala verbs and nouns (Hettige and Karunananda, 2011). The basic operation of Sinhala word formation is joining a word with affixes or other words. There is currently no software tool to implement this operation, or the disjoin operation for morphological analysis.

The objective of this work was to implement a word joining tool for the Sinhala language tools stack.

The functionality of the target tool is summarised by a function *f* that has inputs and outputs as follows:

(combined word, rule) = f(left, right)

Where

1. left and right are valid Sinhala words or morphemes.
2. *combined word* is the valid joined form of left and right. null is also a valid value.
3. rule is the name of the join rule used for the joining when *combined* is not null.

The rest of this paper is organized as follows. We provide a brief introduction to the Sinhala morphology and join rules in section 2. In section 3, we briefly explain the related work done on the areas related to Sinhala morphology and word joining. In sections 4 and 5 we present the challenges faced and our methodology of solving this problem. Finally, in sections 6 and 7, we present our results and conclusions.

2 Sinhala Morphology

Like Sanskrit, Sinhala is rich in inflectional and derivational morphology. In inflection, grammatical forms of a word are formed by applying morphological operations on the lemma (base word). (Karunarathilaka, 1995)

e.g. : *minis* + *u* → *minissu* (මිනිස් + ු → මිනිස්සු)

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 220–226

minis is the lemma of the noun man. *u* is the suffix to generate the plural subject form of the noun.

In derivational morphology, words of different word classes or with different meanings are formed by applying morphological operations on the lemma.

e.g.: *duk* + *pat* → *duppat* (දුක් + පත් → දුප්පත්)

duk means suffering. *pat* means become. The combined word *duppat* is an adjective that means poor.

Based on the two-level morphology concept, these morphological operations can be represented in two stages. (Kimmo, 1984)

1. Lexical Representation
2. Surface Representation

e.g.:
1. Lexical representation of the plural subject form of the noun lemma *minis* is (*minis* + *u)* (මිනිස් + උ)

2. Surface representation of (*minis* + *u*) is *minissu* (මිනිස්සු)

For morphological synthesis, both lexical and surface representation rules should be applied. Surface representation rules are generally based on phonology and may transform both the left- and right-hand morphemes. This transformation is called morphophonemics.

In Sinhala, the most common types of word joining are:

prefix + word → word
word (or lemma) + suffix → word
word + word → word

Where + is the join operator.

Most inflectional morphology operations are of the "lemma + suffix → word" form.

e.g.: *minis* + *u* → *minissu* (මිනිස් + උ → මිනිස්සු)

Many derivational morphology operations are of the "prefix + word → word" and "word + word → word" form.

e.g.: *duk* + *pat* → *duppat* (දුක් + පත් → දුප්පත්)

Meaning is as explained above.

pol + *attạ* → *pollattạ* (පොල් + අත්ත → පොල්ලත්ත)

pol means coconut. *atta* means branch. *pollatta* means the branch of a coconut tree.

In Sinhala, this set of morphophonemic rules are called *sandhi* (join rules). Similar to the *Ashṭādhyāyī* of *Pāṇini* in Sanskrit, The Sinhala grammar book *Sidat Saňgarā* written in 13th century A.D. by *Vēdēha Swāmi* describes some of the grammatical aspects of the Sinhala language.

There are nine join rules in Sinhala language according to *Sidat Saňgarā.*

Following is an example of how the *Sidat Saňgarā* has explained join rules. This join rule is named *Pūrwạ Swạrạ Lōpạ*.

"*pera sarạ lopā parạ sarạ gatạtạ pæminạ*"
(පෙර සර ලොපා පර සර ගතට පැමිණ)

Meaning: Vowel part of the last letter of the left word is replaced by the first vowel in the second word.
According to the above definition, there are two conditions for this rule to be valid.

1. The last letter of the left word must be a combined letter that has a consonant and a vowel part
2. The first letter of the right word must be a vowel.

The other join rules are similarly defined.

2.1 Sinhala Word Join Rules

We represent the join rules described in Sidat Saňgarā in an easily understandable format as follows.
Where

- Ci = consonant (e.g. *k* - ක්)
- Vi = vowels (e.g. *a* - අ)
- Individual letters at the word boundary are written in square brackets. (e.g. [C1])
- Combined letters that have a consonant and a vowel in it is written in [Ci|Vi] form.
  e.g.: *ka* = [*k* | *a*] (ක = ක් + අ)
- L and R are the remaining parts of the joining morphemes.

1. Pūrwạ Swạrạ Lōpạ
L[C1|V1] + [V2]R → L[C1|V2]R

2. Paṛạ Swạrạ Lōpạ
L[C1|V1] + [V2]R → L[C1|V1]R

3. Swạrạ
L[C1] + [V1]R → L[C1|V1]R

4. Swạrādeshạ
L[C1|*a*] + [*i*]R → L[C1|*e*]R
L[C1|*a*] + [*u*]R → L[C1|*o*]R
L[C1|*a*] + [*u*]R → L[C1|*ō*]R

5. Gatrādeshạ
L[C1|V1] + [C2|V2]R → L[C1|V1][C3|V2]R

Where C3 is a member of {*y, v, h, k, t, p, n, m*}

6. Pūrwạ Rūpạ
L[C1] + [C2|V2]R → L[C1][C1|V2]R

7. Gatrākshạrạ Lōpạ
L[*n*] + [C2|V2]R → L[*ňg*|V2]R
L[*n*] + [C2|V2]R → L[*ňb*|V2]R

8. Āgạmạ
L[C1] + R → L[C1|u]R
L[C1] + R → L[C1|i]R

L[C1|V1] + [V2]R → L[C1|V1][C3|V2]R
Where C3 = {*y, v, r*}

9. Dvitvạ Rūpạ
L[C1|V1] + [V2]R → L[C1][C1|V2]R

In addition to the above 9 join rules, we identified a few more join rules in current Sinhala. Some of them are directly taken from Sanskrit and are used in loanwords. The following join rule is an example of a rule that is not in *Sidat Saňgarā*, but currently in use. (Karunathilaka, 1995)

11. Paṛạ Rūpạ

L[C1] + [C2|V2]R → L[C2][C2|V2]R

3 Previous Work

In implementing an English to Sinhala machine translator, Hettige and Karunananda (2011) have implemented a morphological synthesizer. They generate all the forms of all noun classes considering the changes to the letters at the word boundaries in the transformation. They have not used generic joining rules for joining Sinhala words and morphemes but have defined a large number of specific finite state automata to handle multiple letter combination at the word boundaries. However, they do not cover all combinations.

To obtain the indistinct singular subject form of the noun lemma *miti* (short person) and *balu* (dog) they implement 2 different automata, which result in *mittā* and *ballā* respectively.

miti + *ā* → *mittā* (මිටි + ආ → මිට්ටා)
Remove *ti* (ටි) and append *ttā* (ට්ටා)

balu + *ā* → *ballā* (බලු + ආ → බල්ලා)
Remove *lu* (ලු) and append *llā* (ල්ලා)

They have implemented 85 FSA for Sinhala noun formations. However, both of the above transformations use the common join rule called *Dvitvạ Rūpạ*, and may be defined as a single FSA.

Also in their method, the FSA must be input to the noun form synthesizer. For the same letter combinations at word boundaries, different finite state automata must be used for different word morpheme combinations. It is not possible to locate the correct FSA without a comprehensive knowledge of the Sinhala language.

There are no other significant work done in the area of Sinhala morphological synthesis or word joining.

Word joiners have been implemented for other Indic languages such as Hindi and Sanskrit. (Jha et al., 2009; Hyman, 2009; Gupta and Goyal, 2017; Kumar et al., 2010) Some of them use the join rules mentioned in the *Ashṭādhyāyī* of *Pāṇini*. (Jha et al., 2009; Hyman, 2009; Gupta and Goyal, 2017)

Most of them have used finite state transducers to do the morphophonemic operations to obtain the surface form. Most of the morphological tool

implementations for European languages also use finite state transducers to obtain the surface representation for the lexical representation (Lauri et al., 1992). Finite state transducers have been widely used in solving this morphology problem in different language families.

4 Challenges

In Sinhala, for a given pair of morphemes, there may be multiple matching join rules based on the boundary conditions. Also, even when a single join rule is applied, there can be multiple possible outputs, all of which are not necessarily valid for a given pair.

Accordingly, we have identified the following scenarios for a pair of or morphemes.

1. There is only one matching join rule for the pair. The combined form/forms generated by the rule are
    a. valid
    b. partially valid
    c. invalid
2. There are multiple matching join rules for the pair and they yield the same combined form. The combined form is
    a. valid
    b. invalid
3. There are multiple matching join rules and they yield the different combined forms. The combined forms are
    a. valid
    b. partially valid
    c. invalid

In order to detect the correct join rule and the correct join forms accurately, two options were considered.

1. Using another set of rules, which can be applied on top of the standard join rules to eliminate false positives. e.g.: When joining the *naLu* (නළු) and *a* (අ), the *Dvitvạ Rūpạ* join rule is also selected. It generates the form *nalla*. (නළ්ළ). But there is an elimination rule that says the letter *L* (ළ) is not duplicated. Therefore, the form *nalla* is eliminated.

2. Check against the set of all valid Sinhala words, so that you can eliminate invalid Sinhala words.

An attempt was made to collect the language rules that can be used to detect the correct join operation for a given pair. But it was found that the documented secondary rule set is not complete, so that in some scenarios, access to the Sinhala vocabulary is needed to check the validity of the combined forms.

Also, an attempt was made to learn these extra rules from a sample data set with tuples of left, right and combined forms. Sinhala being a low resource language, it is difficult to collect an accurately enriched data set large enough to perform the learning to learn the complete rule set.

Having access to a database of all valid Sinhala words is also not practical. Also there are some valid words generated as combined forms by the join rules, that are not valid combined forms of the given 2 words or morphemes.

Hence a combined solution is proposed. It involves finite state transducers for each join rule and non-tagged corpus of Sinhala words.

5 Methodology

In this research, we implemented a generic Sinhala word joining tool based on the 9 base rules and 4 additional join rules that are currently in use.

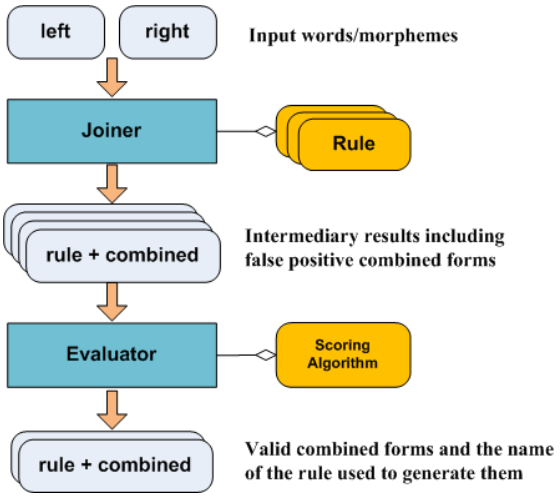


Figure 1: Word joining process

Figure 1 shows the bird's eye view of the word joining process.

Due to the nature of the Sinhala join rules and exceptions in the Sinhala language, finite state

transducers in isolation cannot solve the problem accurately.

For a given left right word/morpheme pair, the joiner applies all the applicable join rules. Some finite state transducers arrive at end states and yield combined forms. All the pairs of combined forms and the join rules used to generate them are returned as intermediate results. There can be both false positives and true positives among them.

A scoring algorithm is introduced to evaluate all the combined forms generated by the join rules. The purpose of the scoring algorithm is to assign a score to each combined form generated by finite state transducers.

The evaluator then selects the results with a score larger than a threshold. The best value for the threshold with respect to a given scoring algorithm is obtained by regressing the joining operation with different threshold values for a sample data set with a manually verified results set.

5.1 Scoring Algorithm

The following parameters are passed to the scoring algorithm.

1. Left most word or morpheme
2. Right most word or morpheme
3. Combined form of the left and right.
4. Name of the join rule used to generate the combined form

The algorithm returns an integer value as the score for the given quadruple.

Our software application uses a corpus and some hand coded elimination rules to derive the score.

It first checks whether the combined form is an invalid joined form of the left and right words or morphemes according to the elimination rules. If it is invalid, the score is set as -1.

If the combined form is not invalid, the word is looked up in the corpus. The occurrence frequency of the word is set as the score. It is a non-negative number.

For the current corpus, the threshold is set as 2. This value has will depend on the size and quality of the corpus. If this set to 0, the number of false positives increases due to the impurities in the corpus. If this is set to a larger value, the number of false negatives increases since the evaluator tends to reject valid combined forms that have a low frequency of occurrence in the corpus.

New scoring algorithms may be plugged-in to the application to obtain better results.

6 Results

The precision and recall were measured for the joining results of the following data sets.

6.1 Dataset 1

8 different grammatical forms of 50 Sinhala nouns were generated by joining their lemma and relevant suffixes. 412 noun forms are expected for the 400 word-morpheme pairs.

6.2 Dataset 2

50 pairs of complete words are joined to generate combined words.

6.3 Precision and Recall

True positives are the correct combined forms for a given pair generated by the application as the end results.

False positives are the incorrect combined forms for a given pair generated by the application as the end results.

False negatives are the expected combined forms for a given pair, but not given by the application as end results. Some of them were eliminated by the scoring algorithm.

	True positives	False positives	False negatives
Dataset 1	401	22	9
Dataset 2	46	0	4
Total	447	22	13

Precision = True positives / (True positives + False positives)

= 447/(447+22)

= 0.9531

Recall = True positives / (True positives + False negatives)

= 447/(447+13)

= 0.9717

7 Conclusion

7.1 False Positives

The analysis showed that the main reason for the false positives is the lack of elimination rules.

E.g.
The input : *ali* + *ā* (අලි + ා)
ali is the lemma of the noun elephant
ā is the suffix to form the singular subject form

Expected output : (*aliyā* - අලියා, *āgamạ*)

Actual outputs : (*aliyā* - අලියා, *āgamạ*), (*allā* අල්ලා, *dwithwạ rūpạ*)

aliyā means elephant. *allā* means god Allah.

The word *allā,* though it occurs frequently in the corpus, is not a valid combined form of the lemma *ali* and the suffix *ā*.

Some false positives are due to the impurities in the corpus.

We may add an exceptions database and introduce more elimination rules to the scoring algorithm to reduce the false positives.

7.2 False Negatives

The analysis showed that the main reason for the false negatives is the incompleteness of the corpus. The occurrence frequencies of the valid combined forms that are not available in the corpus are set as 0. Therefore, they are eliminated by the evaluator.

It is not possible to create a corpus with all valid Sinhala words. Therefore, we may use statistical or machine learning methods to learn further scoring rules.

7.3 Performance

Since our corpus contains 1.2 million entries (including impurities) the database lookup takes a considerable time on test machines. Therefore, an average join operation for a given word pair takes around 20 milliseconds on a 2.5 GHz processor.

7.4 Future Enhancements

A possible future enhancement would be to generate a large sample dataset of tuples of left and right words or morphemes, combined forms and rule name using the current word joiner tool version, get them verified using human input and use that dataset to mine the elimination rules using statistical methods.

The elimination rules mined by this exercise may also be used to implement a scoring mechanism for words that are not available in the corpus.

References

[Department of Census 2012] Department of Census and Statistics Sri Lanka. (2012). Census of Population and Housing. Retrieved from http://www.statistics.gov.lk/PopHouSat/CPH2011/Pages/Activities/Reports/FinalReport/Population/FinalPopulation.pdf

[Geiger 1938] Wilhelm Geiger (1938). A Grammar of the Sinhalese Language. Ceylon Branch of the Royal Asiatic Society (Colombo). Colombo.

[Gupta and Goyal 2017] Priyanka Gupta, and Vishal Goyal (2017), Implementation of Rule Based Algorithm for Sandhi-Vicheda Of Compound Hindi Words. International Journal of Computer Science Issues, vol. 14, no. 2, pp. 45–49

[Hettige and Karunananda, 2011] B. Hettige and A. S. Karunananda. (2011), Computational model of grammar for English to Sinhala Machine Translation. 2011 International Conference on Advances in ICT for Emerging Regions (ICTer)

[Hyman, 2009] Malcolm D. Hyman. (2009). From Pāṇinian Sandhi to Finite State Calculus. Lecture Notes in Computer Science - Sanskrit Computational Linguistics: 253–265

[Ido et al, 1997] Ido Dagan, Lillian Lee, and Fernando Pereira. (1997). Similarity-based methods for word sense disambiguation. Proceedings of the 35th annual meeting on Association for Computational Linguistics

[Jha et al, 2009] Jha G.N. et al. (2009). Inflectional Morphology Analyzer for Sanskrit. In: Huet G., Kulkarni A., Scharf P. (eds) Sanskrit Computational Linguistics. Lecture Notes in Computer Science, vol 5402. Springer, Berlin, Heidelberg

[Karunarathilaka, 1995] W.S.Karunathilaka. (1995). Sinhala Bhasha Vyakaranaya, M.D. Gunasena, Colombo, Sri Lanka

[Kimmo, 1984] Kimmo Koskenniemi. (1984) A general computational model for word-form recognition and production. Proceedings of the 22nd annual meeting on Association for Computational Linguistics

[Kumarathunga, 2000] Munidasa Kumarathunga (2000). Vyakarana Vivaranaya. S. Godage. Colombo, Sri Lanka

[Kumar et al, 2010] Anil Kumar, Vipul Mittal, and Amba Kulkarni (2010). Sanskrit Compound Processor. Lecture Notes in Computer Science Sanskrit Computational Linguistics: 57–69

[Lauri et al, 1992] Lauri Karttunen, Ronald M. Kaplan, and Annie Zaenen. (1992). Two-level morphology with composition. Proceedings of the 14th conference on Computational linguistics

[Murali et al, 2014] N. Murali, R.j. Ramasreee, and K. V. R. K. Acharyulu. (2014). Kridanta Analysis for Sanskrit. International Journal on Natural Language Computing, 3(3):33–49

[Porter, 1980] M.F. Porter (1980). An algorithm for suffix stripping, Program, Vol. 14 Issue: 3, pp.130-137

[Sharma et al, 2002] Utpal Sharma, Jugal Kalita, and Rajib Das (2002). Unsupervised learning of morphology for building lexicon for a highly inflectional language. Proceedings of the ACL-02 workshop on Morphological and phonological learning

Supervised Methods for Ranking Relations in Web Search

Sumit Asthana
Dept. of Computer Sc. and Engg.
IIT Patna, Patna
asthana.sumit23@gmail.com

Asif Ekbal
Dept. of Computer Sc. and Engg.
IIT Patna, Patna
asif@iitp.ac.in

Abstract

In this paper we propose an efficient technique for ranking triples of knowledge base using information of full text. We devise supervised machine learning algorithms to compute the relevance scores for item-property pairs where an item can have more than one value.Such a score measures the degree to which an entity belongs to a type, and this plays an important role in ranking the search results. The problem is, in itself, new and not explored so much in the literature, possibly because of the heterogeneous behaviors of both semantic knowledge base and full-text articles. The classifiers exploit statistical features computed from the Wikipedia articles and the semantic information obtained from the word embedding concepts. We develop models based on traditional supervised models like Suport Vector Machine (SVM) and Random Forest (RF); and then using deep Convolution Neural Network (CNN). We perform experiments as provided by WSDM cup 2017, which provides about 1k human judgments of person-profession pairs. Evaluation shows that machine learning based approaches produce encouraging performance with the highest accuracy of 71%. The contributions of the current work are two-fold, *viz.* we focus on a problem that has not been explored much, and show the usage of powerful word-embedding features that produce promising results.

1 Introduction

Most of the prior works in information retrieval (IR) focuses on retrieving information either using semantic knowledge base or text. In the present day, Information Retrieval (IR) often involves both knowledge base as well as full text search. One cannot succeed in retrieving semantic information with the other.Knowledge base is good at returning precise information, whereas full-text has the benefit of a wide information coverage, for example, Wikipedia articles. Therefore, it is imperative that search uses information from both of the above and tries to find a best approximation.

In our current work we discuss the problem to rank, not entities from a full text search but triples from knowledge bases with the same subject and predicate properties. Let us consider all the professions of a particular person, for example of Arnold Schwarzenegger: *Actor, Athlete, Bodybuilder, Businessperson, Entrepreneur, Film Producer, Investor, Politician, Television Director, Writer*. All of them follow: **"Arnold Schwarzenegger—profession—ProfessionName"**, but some of these are more relevant and prominent whereas others are less. Hence it would be good to come up with a metric to segregate the most-relevant ones' from the less-relevant ones'. The concept of relevance in itself is ambiguous. So here we take the basis as the *amount of information in the Wikipedia article of the entity.*

This type of relevance plays an important role in improving search engines as well as knowledge bases upon which several question-answering systems are being built. For example, all three tasks from the TREC 2011 Entity Track Balog et al. (2011) ask for the lists of entities of a particular type. It is to be noted that ranking of triples using both semantic knowledge base and fulltext articles is not explored at the required level. In order to solve this problem we at first propose models based on supervised machine learning algorithms, namely Support Vector Machine (SVM)

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 227–235

and Random Forest (RF). Therafter, we develop model based on deep Convolutional Neural Network (CNN).

1.1 Related Works

As already mentioned there have not been required number of attempts for ranking triples. The task has been taken up in Bast et al. (2015) with an unsupervised approach. In Cedeño and Candan (2011), authors have proposed an extension to Resource Description Frameowork (RDF) and they called it as Ranked RDF. A ranking model is proposed in Elbassuoni et al. (2009) for SPARQL queries with possible text extensions based on language models. The technique proposed in Dividino et al. (2012) discusses how to combine several kinds of scores associated with triples into a meaningful ranking. In all these frameworks, scores that are similar to our triple scores are assumed to be given.

We start with the approach given in Bast et al. (2015) and come up with new additional features and methods over the existing one. The key contributions of our current work are as follows: (i). we propose supervised machine learning models for triple ranking that exploits both semantic knowledge base and full text information. This is relatively a new direction of research; and (ii). utilizing word embedding information obtained from the Wikipedia knowledge along with the statistical features. Evaluation of the models on WSDM datasets[1] show encouraging performance. (iii) Through the on-going experiments with deep learning based approaches we show that deep CNN can yield promising results for this type of problem.

2 Problem Description and Dataset

The problem that we tackle is related to ranking the relevance of person-profession pairs based on the information present in Wikipedia. This is an example of a non-functional relation between an entity and an abstract group. We have a set of person names and their associated professions from FreebaseBollacker et al. (2008). The goal is to predict a score for each person-profession relation between 0-7, with 7 being the most relevant. A typical set of training examples is:

Wolfgang Amadeus Mozart Composer 7
Wolfgang Amadeus Mozart Pianist 5
Wolfgang Amadeus Mozart Violinist 2

Table 1: Dataset description

Filename	Description
professions	the 200 different professions from professions.kb
professions.kb	all professions for a set of 343,329 persons
profession.train	relevance scores for 515 tuples (pertaining to 134 persons) from profession.kb
persons	385,426 different person names from the two .kb files and their Freebase ids
wiki-sentences[2]	33,159,353 sentences from Wikipedia with annotations of these 385,426 persons
profession.test	relevance scores for 513 tuples (pertaining to 134 persons) from profession.kb

We use the dataset of WSDM cup-2017 triple[1] scoring task, which provides a training and test set comprising of 1,225 person-profession pairs. Details are shown in Table 1. The labels had been obtained via crowd-sourcing wherein 7 independent judges rated each profession for a person as relevant or non-relevant. The scores of these 7 judges were then added to form the composite score described above.

The training sets (the .train files provided above) contain only tuples from the respective .kb files. The person names are exactly the names used by the English Wikipedia. That is, http://en.wikipedia.org/wiki/PersonName takes you to the respective Wikipedia page. For each of the names in persons, there are sentences in wiki-sentences (68,662 sentences for the most frequently mentioned person, 3 sentences for the least frequently mentioned person).

3 Machine Learning based Approach

In this section we describe our proposed approach which starts with defining the problem and then the specific components on word vector generation, feature extraction, query expansion etc.

3.1 Word Vectors

Word embedding (also known as distributed word representations) persuade a real-valued latent semantic or syntactic vector for each word from a

[1] http://www.wsdm-cup-2017.org/triple-scoring.html

large unlabeled corpus by using continuous space language models. Better word representation can be obtained if we have a large amount of training data as the obtained real-valued vectors of words become more representative. We use the popular *word2vec*[3] tool proposed by Mikolov et al. Mikolov et al. (2013a; Mikolov et al. (2013b) to extract the vector representations of words. Owing to its simpler architecture which reduces the computational complexity, this technique can be used for large corpus. We train Word2Vec tool on the **"wiki-sentences"** corpus. The corpus was first preprocessed by removing all numerals, special symbols, and converting to lowercase. The Word2Vec tool was then trained with *feature size of 400, window size of 8, Continuous Bag-of-Word (BoW) model and min count of 15.*

For each profession and person, we generate the word vectors and concatenate to the respective feature vectors of the instances.

3.2 Query Expansion

We treat every given profession word as a topic and apply the query expansion techniques Bast et al. (2015) to expand the profession to a set of 10 most relevant words related to the profession. For example, the profession *Architect* when expanded yields the following set: *architect,design,building,designed,architectural, buildings,church,built,house*.

3.2.1 Logistic Regression

We learn a Logistic regression (LR) classifier for each profession. The positive instances of the classifier denote the Wikipedia articles of persons who only had that profession as mentions and negative samples correspond to the persons who never had that profession as mentions. We obtain this information from Freebase. The LR classifier is trained with the term frequency matrices of the Wikipedia articles of person.We trained one LR classifier per profession giving us a total of 200 LR classifiers, one for each profession. Each such classifier was trained using positive and negative instances created from the Wikipedia articles. The positive instance articles would be articles of people who only had that profession as mentions. The negative articles are articles of people who did not have that profession mention at all. **Profession mention** for both positive and negative instances came from the **persons** file described above which has all possible valid person-profession pairs for the people in the dataset. Thus, the LR classifier for a profession was trained to learn the distinction between a set of articles segregated on the basis of presence/absence of that profession. As a result, the entity of interest out of this training would be the weights the LR classifiers assigned to each word(features).

Looking at the top scoring word (features), it was clear that they were words somewhat distinguishing the positive and negative instances. The LR parameters were tuned using grid search. We only use the classifier if it has an accuracy of more than equal to 80%. We provide link to our query expansion results that are present on github[4].

For other cases, where the LR classifier failed to segregate instances with sufficient accuracy on account of lack of enough data, the method described below was resorted to.

3.2.2 Using word vectors:

Out of 200 professions, about 40 of them (e.g. entertainer) do not have sufficient training data which could lead to a decent accuracy. For such instances, firstly word embedding vectors are created and then top 10 most similar words are retrieved based on cosine similarities. The word embeddings were trained as described above on the wiki-sentences using the gensim toolkit[5]. We use the most_similar function provided by the word2vec model which takes a word and returns the vectors(and associated words) closest to the given word in cosine similarity.

3.3 Features

After query expansion, we extract the following features for each *Person-Profession-Rank* triple. Features are extracted on the Wikipedia articles of the person. Refer to 3.3 for a graphical overview.

1. Word count on full text(*wcFull*) - This feature denotes the count of indicators (most similar words to a profession) and all profession words are present on Wikipedia article of a person.

2. Word count in opening text(*wcOpen*) -This feature corresponds to the count of indicators

[3]https://code.google.com/p/word2vec/

[4]https://github.com/codez266/turnip/blob/master/indicators-pro

[5]https://radimrehurek.com/gensim/models/word2vec.html

Table 2: Scores for methods without word vectors(accuracy represents exact matches)

Method	Accuracy(δ=0)	Average Score Difference	Kendall's Tau
Counting(Baseline)	0.68	1.92	0.42
SVM	0.69	1.86	0.37
Random Forest	0.71	1.80	0.34

Table 3: Scores for methods(accuracy represents exact matches)

Method	Accuracy(δ=0)	Average Score Difference	Kendall's Tau
Counting(Baseline)	0.69	1.90	0.39
SVM	0.70	1.86	0.35
Random Forest	0.71	1.78	0.33

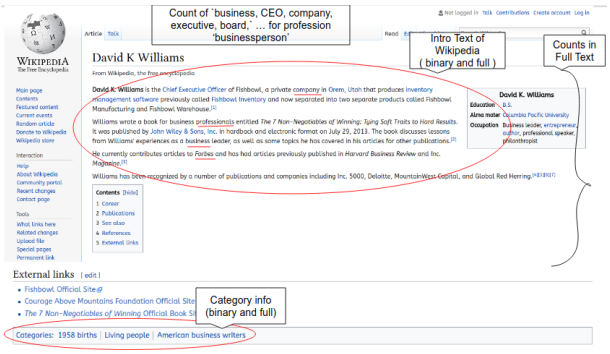


Figure 1: Features on Wikipedia article

and all profession words present in the introduction text of Wikipedia article of person.

3. Word count in Category(*catCount*) -This indicates the count of all profession words in the category section of Wikipedia article.

4. Binary presence in full text(*catBin*) - This denotes the presence or absence of profession words in the category section of an article.

5. Presence in opening text(binary)(*bOpen*) - This feature denotes the presence or absence of all profession words in opening text of Wikipedia article of person.

6. (*wVec*) - This feature is defined based on the word embedding vectors as defined earlier. It is obtained by concatenating the word vectors of a person name and profession name. E.g. for *Wolfgang Amadeus Mozart Composer 7* we concatenate vectors of **"Wolfgang Amadeus Mozart"** and **"Composer"** to form an 800 dimension vector and add it to the existing vector of other features as described in this section.

Finally, we input a feature vector of dimension 805 to the classifiers.

3.4 Justification for using the additional word vectors as features

As word2vec(Mikolov et al., 2013b) mentions learning information about various features of words with respect to their context, this information in encoded in the dimensions of the vector. We intend to use this dimensional information as an input to the classifier so check if the contextual information contains some signal to distinguish professions or not.

3.5 Classifiers

We develop models using three classifiers. We use the scikit-learn library for implementation of these classification models. The grid-search[6] module was used to optimize the above set of parameters and get the best performing set.

1. Scores based on just the normalized raw counts of words for professions associated with a person. This method formed the baseline and as such did not use any classifier. The scoring was done based on normalizing raw counts across person-profession pairs for the same person.

2. SVM classifier which is developed with the above set of features(Sec 3.3).

3. Random Forest classifier developed with the above set of features(Sec 3.3).

For the last two cases, the instances with score 0-3 are mapped to label 0 and instances with score 4-7 are mapped to 1. During testing, binary output of

[6]http://scikit-learn.org

the classifier is projected in a similar manner to get the final scores. The label prediction 0 is mapped to 0-3 using the normalized raw counts as per the first approach. The label prediction 1 is mapped to 4-7 using the same approach. An example of using normalized raw counts to generate scores: For a person X, consider the professions with raw counts of associated words:

- Actor - 20
- Director - 10
- screenwriter - 7

As per the information, actor would get a rating of 6-7, director would be scored as 3-4 and screenwriter roughly 2-3.

The **reason** for adopting this hybrid approach was that final rankings had to be from 0-7 which reflected the degree of belongingness of profession to the person. However, this is a very fine-grained scoring for a classifier and a ranking of 2/3 or 4/5 isn't much different. If we had used seven different labels, the classifier would have tried to draw a fine decision boundary across all seven classes, which isn't feasible. Therefore, we thought it best to use a classifier to segregate between relevant and non-relevant, and then adjust in a post-processing normalization step to generate scores with the dataset requirements(i.e between 0-7). This also has the benefit of being close to how the users rated the person-profession pairs. (Bast et al., 2015) mentions that each user chose between relevant and non-relevant when presented with the person-profession pair during the training step.

We now move on to provide details about the classification.

1. ***Counting Approach: Baseline***: The baseline model that we define is based on counting profession word and its indicators in the article of the person whom we have to rate. Only profession word is not always indicative of the actual person-profession relation. For example, let us consider, *"Jolie made her screen debut as a child alongside her father"*. Here, **"screen"** or **"debut"** somewhat convey an acting profession. Hence, this observation necessitates the need for finding more relevant words (i.e. indicator words) related to a profession, which we also include in the counting alongwith the main profession word. This is the simplest approach which involves counting the profession and its indicator words and normalizing them linearly, with the greatest of them achieving the score 7 and accordingly.

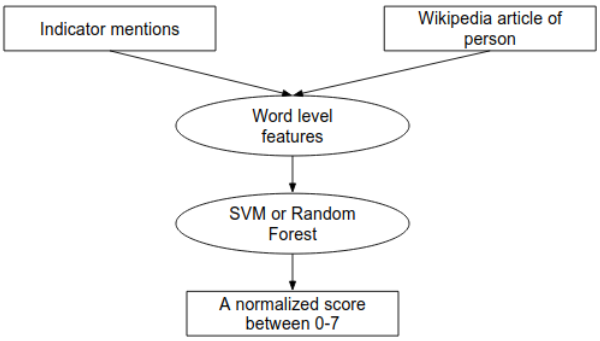


Figure 2: Pipeline for Classification

2. ***SVM based Approach***: With a combination of the features as described above, SVMs (Joachims, 2002) are trained to learn the relevance. Grid search is used to the tune given parameters:(Best - *Kernel: rbf, C: 1, Gamma: 1*)
   - Kernel: *rbf, linear*
   - C(Penalty Parameter): *0.01,0.1,1,10,100*
   - Gamma(Kernel Coefficient): *0.001,0.01,0.1,1,10*
3. ***Random Forest based Approach***: Similar to SVMs, we use the same set of features to learn Random Forest (Breiman, 2001). The grid search parameters are set as:(Best - *max_features: sqrt, n_estimators: 10, min_samples_split: 0.05*)
   - oob_score: *True*
   - max_features: *sqrt, log2*
   - n_estimators: *10, 100*
   - min_samples_split: *0.05, 0.10, 0.15, 0.20*

3.6 End to end pipeline

The relevance scoring mechanism consists of the following stages: Fig. 2 shows the basic way of training the classifier.

1. Indicator words generation for professions for which enough data is available in the form of articles of people in that profession. This

step uses learning an LR classifier per profession as described earlier. In parallel, we use word2vec to generate indicator words for the less prominent professions.

2. Each training instance is a person-profession-rank triple and the test instance a person-profession pair. We use the Wikipedia article of the person and the set of 10-15 indicator words so generated along with the original profession words to generate the feature values on the article. We get a feature vector of length five from this. We append the additional 800 dimensional vectors generated through word embedding (by Word2vec tool) for the profession and person in each instance. This produces a resulting vector of 805-dimension.

3. This vector is fed to the classifier discussed above, which was trained to do a binary classification of relevant/non-relevant.

4. These binary classification labels from the above classifier were then scaled to the values between 0-7 (discussed in the introduction of classifiers section) to conform to the output standard for analysis.

4 Experiments and Analysis

4.1 Evaluation

We use three metrics to measure the efficiency of the baseline and the proposed models.

1. Accuracy - The percentage of person-profession triples that matched.

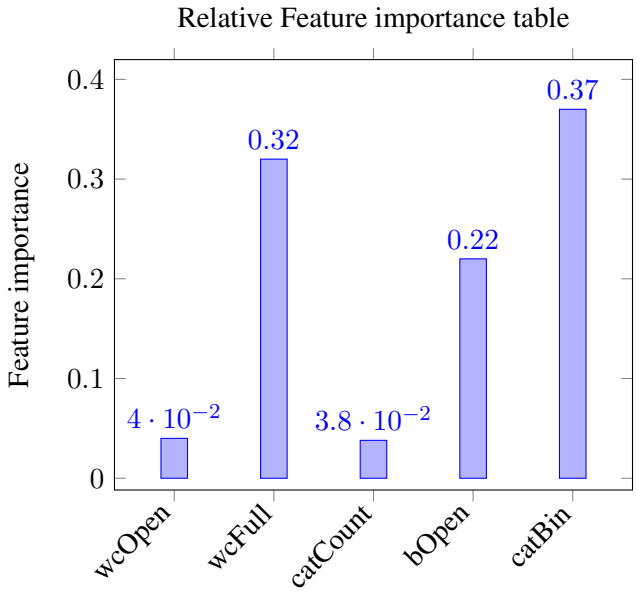
2. Kendall's tau - $\tau_p = 1/Z(n_d + p.n_t)$ where n_d is the number of discordant (inverted) pairs, n_t is the number of pairs that are tied in the gold standard but not in the predicted ranking or vice versa, p is a penalization factor for these pairs which we set to 0.5, and the normalization factor Z (the number of ordered pairs plus p times the number of tied pairs in the gold standard). This is to account for the tied rankings in the gold standardFagin et al. (2004).

3. Average Score Difference - Average of the difference of scores between gold mention and predictions.

4.2 Scores and Best Features

We perform 10-fold cross validation on the training data for optimizing the model and evaluate on the the test set. **Table 2** shows the scores for SVM and Random Forest along with the baseline. It shows that random forest based model performs slightly better than SVM. However, both of these approaches perform better compared to the baseline model.The 800-dimension concatenated word vector of person and profession did not provide generalization as we expected. A possible reason could be the insufficient size of **wiki-sentences** with only 33000k sentences, which were used for training of Word2Vec tool. However it is to be noted that that word embedding vectors were more useful for generating the indicator words of a profession.

We measure the importance of each feature and its effect (except word vectors). Importance to these features is extracted using **feature_importances_** data structure provided by scikit-learn after training them.



Clearly, **wcOpen**(word count in opening text) and **catCount**(word count in category) do not have convincing roles. A possible reason for this might be that:

1. *Opening text* of Wikipedia and *category* do not often enumerate all the professions.

2. Their relative counts in a small paragraph are not sufficient.
   The latter observation is backed by the fact that binary word presence in the first para

(**bOpen**) is a good feature where we only account for the presence or absence of a profession word.

3. As expected, the full-text search of profession words along-with its indicators is a significant feature.

4. The most important thing is the importance of the binary feature of profession word in **category catBin**, which shows that Wikipedia categories are the reflection of subject matter in a comprehensive manner.

4.3 Error Analysis

We perform a thorough analysis to understand the shortcomings that still need to be tackled:

1. *Popularity*: The way human ranked the persons is not very well defined and is hugely affected by popularity. Popular personalities get ranks for professions based on a lot more prior knowledge than just a Wikipedia article. Unpopular personalities are assigned ranks based on what is directly visible in the initial glance of the Wikipedia article, in most cases.

2. *Amount of information*: The ranks were also affected by the amount of information present in the Wikipedia article of a person. For example, **Ba. U** has been rated as **2 for politician** and **7 for Lawyer**, although he has been actively involved in politics, having been the president two times. The only issue is that his Wikipedia article is too short to provide a substantial information.

3. *Drawback of a linear relationship based on word count*: It is clear that the underlying idea of indicator word count is not so much useful. Often for very long articles the count seems to lose its meaning. For example, **Napoleon** has been rated as **7** for both **politician** and **military officer**, but given his long description of military campaigns, military officer seems to outweigh politician during prediction.

4. *Experiments with Word Vectors* The word vectors were trained on the wiki-sentences to get the context from the Wikipedia mentions. The use of word embedding vectors improves the accuracy to some extent, and greatly helps in deriving more contextual information for 25% profession words for which we had very less person mentions from Freebase. This shows that they can be used in places where we have insufficient information in semantic space to derive context.

4.4 Comparison with previous work

We'd like to mention that the previous work (Bast et al., 2015) achieved an overall accuracy of 63% with their method **MLE combined**. We achieved an overall accuracy of about 70%. However, they do better on the average score difference front, getting 1.57 as best with the **Count Combined method**. We report the best average score difference as 1.78 with Random Forests and word vector features. Kendall's tau is 0.22 for them with **MLE combined** whereas its 0.33 for us with random forests. However, we mention that our training data was significantly less than (Bast et al., 2015) because they used the entire Wikipedia for training whereas we only used wiki-sentences for word vector generation and individual Wikipedia articles of persons for feature generation per instance.

4.5 Relation to web-search

We mention **Web search** in the title because getting relevance scores for several entity relation pairs with different predicates but same entity can help to show only the most significant one's in cases where only one or two results are required. This is especially useful for filtering in today's world of search where search engines also take information from knowledge bases like Wikidata[7].

5 Deep Learning based Approach

We exploit deep learning algorithm being motivated from the fact that it does not require any feature engineering. For our deep learning based approach, we develop a model based on CNN. CNNs are able to convolve over the entire text just like on images and are able to extract features from the text efficientlyKim (2014). We present the results achieved till now using CNN to identify single relation entities, e.g people with only a single profession. Though not complete, this step is important as it can be extended to the multi-profession case. The final ranking generation for

[7] https://www.wikidata.org/

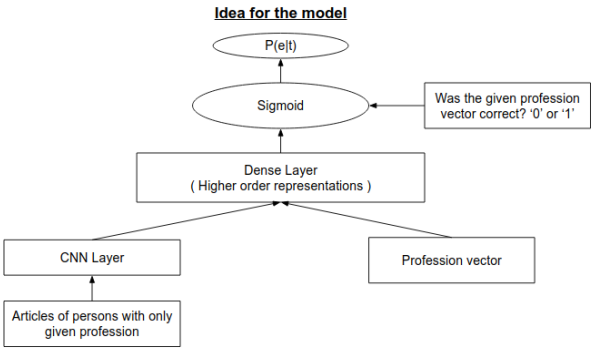


Figure 3: Intuition for CNN

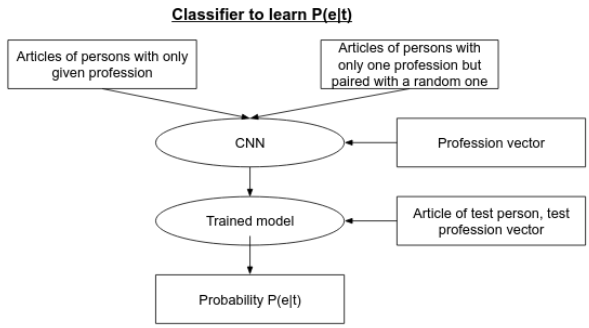


Figure 4: Deep network classifier for profession

multi-relation entities using CNNs is left as a followup of this work or can be taken up anywhere else.

Unlike the previous model, which had a different classifier for identifying indicator words for each profession, here we use a single deep network classifier to find $P(e|t)$ where **'e'** is the person entity and **'t'** is the type(profession), i.e, we find the degree of belongingness of the entity to the type.

Fig. 4 shows the basic way of training the classifier which is quite similar to one described in the previous approach. We use both Convolution Neural Network (CNN) and Long Short Term Memory (LSTM), but we found better results with CNN and hence report the results using only this classifier.

The only difference is that we now use a single classifier to classify across all person-profession pairs and we now also include profession information as a word vector of profession. The underlying idea is to learn *common high level representations that make a person and a profession similar*.

Fig. 4 shows the basic way of training the classifier.

Fig. 3 shows the basic overview of the CNN model.

Table 4: CNN statistics

Samples	22638
Hidden Layers	2
Hidden Neurons	512, 64
Embedding Dimension	300
Precision	82%

Parameters to the CNN classifier:

- Google-News word vectors[8] of dimension 300.
- 50 filters of size 3 each and one convolutional layer.
- GlobalMaxPooling and AveragePooling but found that MaxPooling performed better in all cases.
- Two dense layers after convolution, which were formed after **merging convolved Wikipedia article of person and profession vector** Fig. 3.
- The first dense layer has **512** neurons and second one has **64** neurons.

Table 4 shows statistics using CNN classifier which was implemented using keras[9]. For efficient scoring, we considered only first two thousand letters in the Wikipedia article of each person.

It was found that CNN based classifier performs very well while classifying single-profession entities as correct/incorrect pair, but when extended to multi-profession entities it was somewhat not able to distinguish the more relevant professions from the less relevant one's. We attribute this to the noise introduced by several professions in the Wikipedia article of the person and leave this as an interesting task to explore as a follow-up of this work, or elsewhere.

6 Conclusion

In this paper we have proposed supervised machine learning based solutions for handling triple ranking in a mixed domain of knowledge base and

[8] code.google.com/p/word2vec/
[9] keras.io

full-text. We have explored two supervised classifiers with handcrafted features extracted on English Wikipedia along with word embeddings to learn the rankings. Some of the features work well to learn the rankings but more can be explored. Moreover, by using CNN as a classifier to learn representations of person-profession entities, we have shown that deep learning can be applied to this domain and provide and interesting alternative method to explore further.

References

Krisztian Balog, Pavel Serdyukov, and Arjen P. de Vries. 2011. Overview of the TREC 2011 entity track. In *Proceedings of The Twentieth Text REtrieval Conference, TREC 2011, Gaithersburg, Maryland, USA, November 15-18, 2011*.

Hannah Bast, Björn Buchhold, and Elmar Haussmann. 2015. Relevance scores for triples from type-like relations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 243–252. ACM.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*, pages 1247–1250. ACM.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Juan P. Cedeño and K. Selçuk Candan. 2011. R^2df framework for ranked path queries over weighted RDF graphs. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics, WIMS 2011, Sogndal, Norway, May 25 - 27, 2011*, page 40.

Renata Queiroz Dividino, Gerd Gröner, Stefan Scheglmann, and Matthias Thimm. 2012. Ranking RDF with provenance via preference aggregation. In *EKAW*, volume 7603 of *Lecture Notes in Computer Science*, pages 154–163. Springer.

Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum. 2009. Language-model-based ranking for queries on rdf-graphs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 977–986.

Ronald Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, and Erik Vee. 2004. Comparing and aggregating rankings with ties. In *PODS*, pages 47–58. ACM.

T. Joachims. 2002. *Learning to Classify Text Using Support Vector Machines – Methods, Theory, and Algorithms*. Kluwer/Springer.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751. ACL.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Malayalam VerbFrames

Jisha P Jayan
Centre for Development of Imaging Technology
Thiruvanathapuram
jishapjayan@gmail.com

Asha S Nair
Centre for Development of Imaging Technology
Thiruvanathapuram
ashanaircdit@gmail.com

Govindaru V
Centre for Development of Imaging Technology
Thiruvanathapuram
neithalloor@gmail.com

Abstract

Verbs acts as a major role in describing a sentence meaning. Capturing of the syntactic distributions of occurrence of a verb in a sentence is the VerbFrame. This paper tests the applicability of verbframe approach that has been developed for Hindi language in Malayalam. Around 255 verbs were selected for this study, showing the basic argument structure of words with these verbs.
Keywords- verbframe; karaka relations; semantic; syntactic;

1 Introduction

Verbs are the most important grammatical category in any language. With the help of an action, activity and state are denoted. The arguments of the verb indicate various participants required by the verb. Verbs play a noteworthy part in interpreting meaning of a sentence, therefore, the study of the argument structure of a verb and their syntactic behavior will provide the needed knowledge base for intelligent NLP applications. Verbframe is the gathering of the syntactic distribution of the verb occurrence in any sentence. Paninian Grammatical Framework (PGF) is followed in creating a Verbframe as verb plays the important role in the sentence analysis.

The relation of verb with the alternate units of a sentence, in a language may be encoded in various ways. Among them, the word order and the presence of case markers on the arguments are very often used by computational linguists. There are, however, languages in which the marking can be present of the verb itself rather than its arguments (Butt, 2010). Such types of relations frequently reflect semantics of a verb, that-means the syntactic behavior of the verb provides a good support to understand its semantics. Researchers also encode other information such as tense, aspect, modality, gender, number, person etc., with verb, that allow language specific variations.

This paper is intended to develop verbframe for Malayalam language which has got grammatical roots from Dravidian and Aryan languages. This paper presents the work in different stages, beginning in Section 2 with the major works related. Section 3 introduces the Verb Frame and its description . Section 4 describes the Verb frame for Malayalam. Finally, Section 5 concludes the paper.

2 State of Art

Some of the famous linguistic sources related to verb argument structure, are discussed briefly in this section. Levin's work on verb classes (Beth, 1993) indicates the relationship between semantic and syntactic behavior of the English verbs. The verb behavior can be used to get an insight into linguistically applicable aspects of the verb meaning (Beth, 1995). VerbNet (VN) (Kipper, 2000) (Kipper, 2005) is a domain-independent; hierarchical, wide-coverage of online verb dictionary which extends Levin's verb classes (Beth, 1993) and providing syntactic and semantic information for English verbs. It is mapped to various language resources such as Wordnett (Fellbaum, 1998), FrameNet, and PropBank. Each class of verbs in VN is described by thematic roles, selectional restrictions on the arguments, and syntactic frames (Beth, 1993).

PropBank (PB) (Palmer, 2003) (Palmer, 2005) is a corpus, annotated with verbal propositions and their arguments. This has been extensively used for semantic role labeling task in recent times (CoNLL shared task 2004-05 and 2008-2009). PB gives a layer of semantic annotation upon the syntactic structures. PB represents the verb argument depending on the valency of the verb relations by Arg0, Arg1, Arg2, etc., (Palmer,

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 236–244

2002). Each set of argument labels and their definitions is called a frameset. As an example, consider the frameset for the verb *dance*. This verb takes the dancer:Arg0, dance:Arg1 , partner:Arg2 and audience:Arg3 as essential roles. It also has non-essential roles such as location:Argm-loc and time:Argm-tmp. This is for capturing spatio-temporal aspects of verbs.

FrameNet (FN) (Baker et al., 1998) is an online lexical resource for English, based totally on frame semantics and supported by means of corpus evidence. FN groups words in accordance to the conceptual structures, i.e., frames that underlie them (Arun, 2008) . The paper describes three major components such as: (1) Lexicon; (2) Frame Database; (3) Annotated Example Sentences. The Frame database deals with the descriptions of each frame's basic conceptual structure, and provides the names and descriptions of the elements participating in such structure (Begum, 2017). Annotated Sentences are marked to illustrate the semantic and morpho-syntactic properties of the lexical items. Each frame contains numerous elements, i.e., core (core arguments) and non-core (adjuncts or peripheral roles) elements which are considered as semantic roles. For example, core elements of the frame Getting-up are person/animal getting up from sleep and place of sleeping; non-core elements are time, purpose, etc.

All these resources looks into the argument structure of English verbs. They gives the syntactic and semantic information, and correlation between them. These resources are also mapped to each other making individual resources much richer. In the work of creating verb frames for Hindi, the argument structure of verb is captured using Karaka relations which capture both syntactic and semantic information about the verbs. Between Karaka relations, thematic roles and Propbank annotation, a mapping is done . Begum et al. (Begum, 2008) mentioned their experience with the creation of Hindi verb frames. These frames are further classified based on a Paninian grammar framework using 6 Karaka relations. This method considered the morphology, syntactic variations and semantics of the verb to divide it into various classes.

Based on similar approach, Ghosh (Ghosh, 2014) created a resource for verb frames for compound verbs in Bengali language. The main aim of the paper is to investigate if the vector verb from the compound verb is able to retain its case marking properties and argument structure or not. Additionally the knowledge and syntax associated with verb frames can be utilized for categorizing and analyzing the verb words for various NLP applications.

Soni et al. (Ghosh, 2013) explores the application of verb frames and the conjuncts in sentence simplification for Hindi language. The method proposed by the authors includes usage of conjuncts as a first level of sentence simplification. This is followed by using verb frames enhanced with tense, aspect and modality features. It is a rule based system and its output is evaluated manually and automatically using the BLEU score for the ease of readability and simplification.

A semi-automatic annotator tool for verb frames was developed by Hanumant et al (Redkar, 2016). The tool is used for extracting and generating the verb frames automatically from the example sentences of Marathi wordnet. The paper explains the concept and working of the verb - frame tool with its advantages and disadvantages. Other related work by Schulte (Walde, 2009) has also explored verb frames for the English language.

3 Verb Frames

In all languages, verb plays the major part-of-speech category. Verbs are used to define actions, activities and states. Ability of the verbs to choose their arguments and/or adjuncts is termed as 'verb sub-categorization'or 'verb valency'. Combination of functional units that are elicited by a verb is refered to as verb frames. In linguistics, verb-framing and satellite-framing are typological descriptions of how verb phrases in different languages describe the path of motion or the manner of motion, respectively (Redkar, 2016).

Verb frame generally constitutes verbal propositions and arguments of words surrounding a verb in a given sentence. Each of the prepositional words in a verb frame has arguments such as an arc-label, otherwise called a semantic role label, its necessity in a frame, case markers or the suffixes, lexical type, relation of the word with head verb, position with respect to head verb, etc. These verb frames are developed to generate dependency tree structures in a given language. Verb frames on the basis of their argument demands categorization of any verb. The verb frames show mandatory

Karaka[1] relation for a verb. They are:

1. *Karaka : dependency arc labels.*
2. *The necessity of the argument whether it is mandatory (m) or desirable (d).*
3. *Case Markers / Vibhakti: post-position or the case associated with the nominal.*
4. *Lexical category of the arguments.*
5. *The Position of the demanded nominal with respect to verb whether it is left(l) or right(r).*

Verb frames are built for the base form of a verb. The demands undergo a subsequent change based on the tense, aspect and modality (TAM) of the verb used in the sentence. Knowledge about the transformations induced on the base form of a verb by TAM is stored in the form of transformation charts for each distinct TAM.

In the present work we develop verbframe for Malayalam based on Karaka theory developed by IIIT-Hyderabad for Hindi.

4 Malayalam Verb Frame

Amid the semantic analysis, verb is taken as the central, element of the sentence. According to Paninian viewpoint, there are four levels in understanding any sentence (Bharati, 1995) namely the surface level (uttered sentence), the vibhakthi level, the Karaka level and the semantic level. The Karaka level has related to semantics on one side and on the other side with the syntax. Karaka relation can be identified from markers/suffixes or case endings after the noun. The Karaka relations in Malayalam are analyzed from the point of vibhakthi and the postpositions that associate with it. The types of verb and the vibhakthi markers in Malayalam are illustrated in Figure 1 and Table 1 respectively.

The roles and the dependency relation based on IIIT_H approach, are shown in Table 2.

The genitive noun does not have any direct grammatical or semantic relation with the verb but only the noun modified by the genitive is related to the verb. The Genetive case "സംബന്ധികാവിഭക്തി" *saMbhndhikaavibhakti* otherwise Possessive takes the markers "ന്റെ" *nRe*, "ഉടെ" *uTe*.
Eg (a): രാമന്റെ അനിയൻ വന്നു.

[1]karakas are the typed dependency labels in Computational Paninian Framework (Bharati, 1993)

No	Case	Case markers
1	nirddeeSika നിർദ്ദേശിക Nominative	ϕ
2	prathigraahika പ്രതിഗ്രാഹിക Accusative	എ -e
3	samyojika സംയോജിക Sociative	ഓട് -ooTu
4	uddeeshika mal Dative	ക്ക്, ന് kku ,nu
5	sambandhika സംബന്ധികാ Gentive	ന്റെ , ഉടെ -nRe,-uTe
6	aadhaarika ആരാധിക Locative	ഇൽ, കൽ -il, -kal
7	prayoojika പ്രയോജിക Instrumental	ആൽ aal
8	sambhoodana സംബോധിക Vocative	ീ , ാ, േ long forms
9	മിശ്രവിഭക്തി Ablative	ഇൽ നിന്ന് il ninn

Table 1: Case and Case Markers

ramanRe aniyan vannu.
Raman's brother came.
Eg (b): അവളുടെ അമ്മ പറഞ്ഞു .
avaLuTe amma paranjnju.
Her mother said.
Because of this, the genitive noun can be removed from the sentence without affecting the grammaticality of the sentence

Dependency annotated data are used for developing Malayalam verb frames. The dependency annotation is a collective process of Tokeniser, Morphological Analyser, POS tagger, Chunker and Dependency annotation. A raw text will be given as the input and the text is converted into tokens, identifies grammatical features of the individual words, assigns parts of speech (POS) tags to each word , groups them to phrases and the dependency tree diagrams are drawn. Malayalam has tendency to join a wide variety of suffixes with a single word forming compound words, which makes the process more complicated. Therefore complicated words are spllited and then analysed in the present analysis. As an example, consider the following sentence.
മറ്റ് ഭക്ഷണ സാധനങ്ങളെ അപേക്ഷിച്ച് പഴങ്കഞ്ഞിയിൽ ബി6 , ബി12 വൈറ്റമിനുകൾ ധാരാളം അടങ്ങിയിട്ടുണ്ട് .
maRRu bhakshaNa saadhanangngaLe apeekshiccu pazhangkanjnjiyil b6 , b12 vaiRRaminukaL

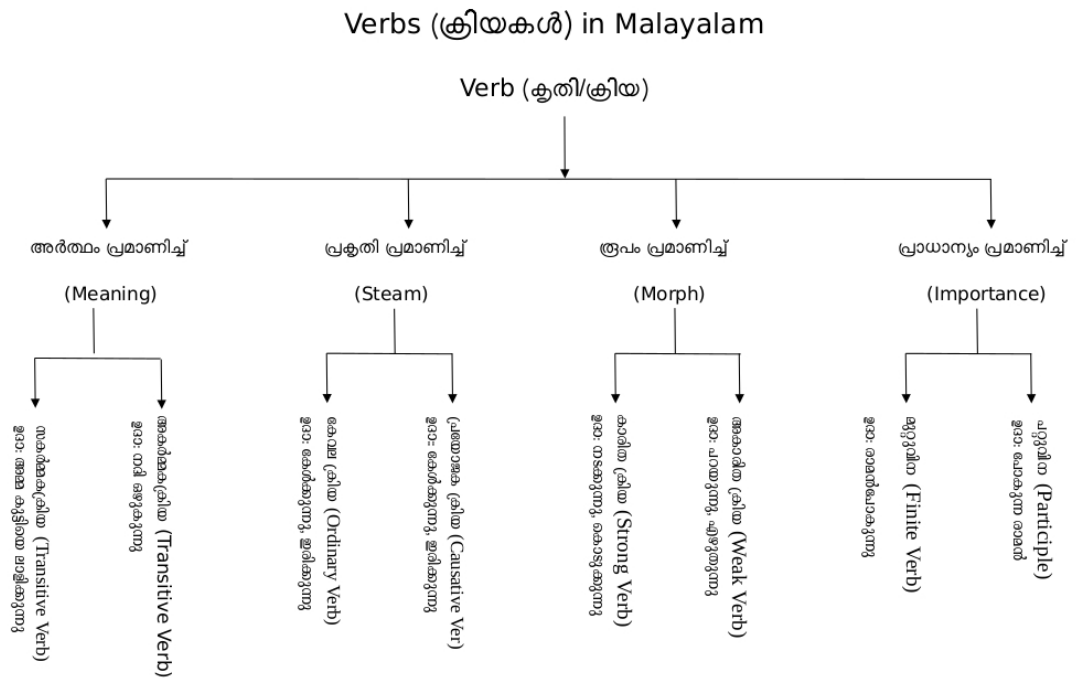


Figure 1: Verb types in Malayalam

dhaaraaLaM aTangngiyiTTuNTu.

In comparison to other food items, rice gruel is rich in vitamins B-6 and B-12.

The sentenence is annotated as follows:

<Sentence id="1">

1 ((NP <fs af='സാധന,n,ne,pl,3,d,എ,NGaLe' head='സാധനങ്ങളെ' name='NP' drel='k2:VGF'>

1.1 മറ്റ് QT_QTF <fs af='മറ്റ്,qtf,,,,,,' name='മറ്റ്'>

1.2 ഭക്ഷണ JJ <fs af='ഭക്ഷണ,adj,,,,,,' name='ഭക്ഷണ'>

1.3 സാധനങ്ങളെ N_NN <fs af='സാധന,n,ne,pl,3,d,എ,NGaLe' name='സാധനങ്ങളെ'>

1.4 അപേക്ഷിച്ച് PSP <fs af='അപേക്ഷിച്ച്,psp,,,,,,' name='അപേക്ഷിച്ച്'>

))

2 ((NP <fs af='പഴങ്കഞ്ഞി,n,ne,pl,3,d,ഇൽ,il' head='പഴങ്കഞ്ഞിയിൽ' name='NP2' drel='k7:VGF'>

2.1 പഴങ്കഞ്ഞിയിൽ N_NN <fs af='പഴങ്കഞ്ഞി,n,ne,pl,3,d,ഇൽ,il' name='പഴങ്കഞ്ഞിയിൽ'>

))

3 ((NP <fs af='ബി6,n,ne,sg,3,d,0,0' name='NP3' drel='ccof:NULL_CCP'>

3.1 ബി6 N_NN <fs af='ബി6,n,ne,sg,3,d,0,0' name='ബി6'>

3.2 , RD_PUNC <fs af='&comma,punc,,,,,,' name=','>

))

4 ((NULL_CCP <fs af=',,,,,,,' dmrel='k1:VGF' name='NULL_CCP'>

4.1 NULL CC <fs af=',,,,,,,' name='NULL'>

))

5 ((NP <fs af='വൈറ്റമിന്,n,ne,pl,3,d,കൾ,kaLu' head='വൈറ്റമിനുകൾ' name='NP4' drel='ccof:NULL_CCP'>

5.1 ബി12 N_NN <fs af='ബി12,n,ne,sg,3,d,0,0' name='ബി12'>

5.2 വൈറ്റമിനുകൾ N_NN <fs af='വൈറ്റമിന്,n,ne,pl,3,d,കൾ,kaLu' name='വൈറ്റമിനുകൾ'>

))

6 ((JJP <fs af=',qtf,,,,,,' head='4' name='JJP' drel='pof:VGF'>

6.1 ധാരാളം QT_QTF <fs af='ധാരാളം,qtf,,,,,,' name='ധാരാളം'>

))

7 ((VGF <fs af='അടങ്ങ്,v,,,,,ഇഉണ്ട്,iTTuNTu' head='അടങ്ങിയിട്ടുണ്ട്' name='VGF' Participles_m='ഇട്ട്' Participles='yes'>

7.1 അടങ്ങിയിട്ടുണ്ട് V_VM_VF <fs af='അടങ്ങ്,v,,,,,ഇഉണ്ട്,iTTuNTu' name='അടങ്ങിയിട്ടുണ്ട്'

Karakas	Case	Case Marker		Role
Karthru Karakam	Nominative	ϕ	k1	Agent/Subject/Doer
Karma Karakam	Accusative	e എ	k2	Object /Patient/Causer
Karna Karakam	Instrumental	aal ആൽ	k3	Instrument
Sampradana Karakam	Dative Sociative	kku ,nu (ക്ക്, ന്) ooTu(ഓട്)	k4	Recepient/Experiencer
Apadana Karakam	Ablative	il ninn ഇൽ നിന്ന്	k5	Source
Vishayadhikarana Deesaadhikarana Kaladhikarana	Locative	ഇൽ il	k7 k7p k7t	Locative (in general) Location in space Location in time

Table 2: Karakas and Role (IIIT-H)

Participles_m='ഇട്ട്' Participles='yes'>
))
8 ((BLK <fs af='.,punc,,,,,,' head='.36' name='BLK' drel='rsym:VGF'>
8.1 . RD_PUNC <fs af='.,punc,,,,,,' name='.'>
))
</Sentence>

The dependency annotation is tree is given in Figure 2.

4.1 Diagnosis

Diagnosis of Malayalam verbfame is illustrated with an example of a verb entry with the description and verb frame. Gloss explains meaning of the particular verbal root. Arc label is to show the dependency relation between any words and the verb that exist in a sentence. Necessity is valency. Valency is the the number of grammatical aspects of verbs which combines other words in that sentence. On the other hand, it is the capacity of verbs that how many arguments, it can combine with itself at time. The distinction among the modifiers and complements is mostly defined using valency, which is a central notion in the theoretical tradition of dependency analysis (Theoritical tradition of dependency analysis has limitation in Computational Linguistics that has discussed widely in recent years. So it is not detailed here). Although the exact characterization of this notion differs from one theoretical framework to the other, valency is usually related to the semantic predicate-argument structure associated with certain classes of lexemes, in particular verbs but sometimes also nouns and adjectives (Nivre, 2005). The idea is that the verb imposes requirements on its syntactic dependents that reflect its interpretation as a semantic predicate. Dependents that correspond to the arguments of the predicate can be mandatory or optional (Nivre, 2005). The valency frame of the verb is generally taken to incorporate argument dependents, however some theories also allow desirable non-arguments to be included. Position basically refers to on which side of the verb, the particular word takes place in the sentence. That is, the word can occur either on the left side of the verb or right side of the verb. So, 'l' stands for the word left and 'r' stands for the word right.

Verb::അടങ്ങ് *aTangngu*
SID:: അടങ്ങ് %VT%S1
Verb Sense::
Eng_Gloss::to contain
Verb Class::
Verb_in_Same_Class::
TAM for the verb root::ഇഇട്ട്ഉണ്ട് *i iTT uNTu*

Frames::
Example::മറ്റ് ഭക്ഷണ സാധനങ്ങളെ അപേക്ഷിച്ച് പഴങ്കഞ്ഞിയിൽ ബി6 , ബി12 വൈറ്റമിനുകൾ ധാരാളം അടങ്ങിയിട്ടുണ്ട് .
maRRu bhakshaNa saadhanangngaLe apeekshiccu pazhangkanjnjiyil b6 , b-12 vaiRRaminukaL dhaaraaLaM aTangngiyiTTuNTu.
In comparison to other food items, rice gruel is rich in vitamins B-6 and B12.
FRAME_ID::1

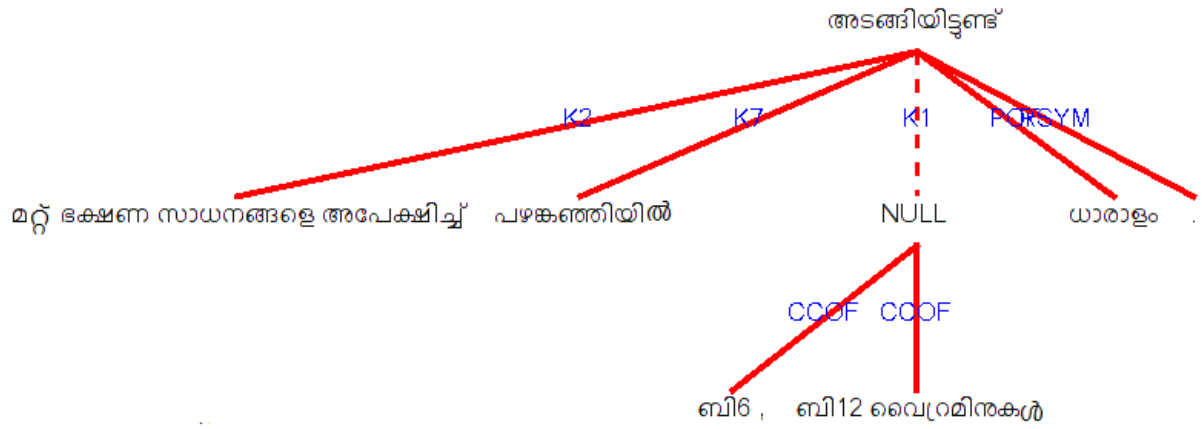


Figure 2: Dependency tree for given example

arc_ label	nec- essity	Vibhakti	Lex Type	posn	reln
k2	m	കൾഎ (kaLe)	n	l	c
k7	m	ഇൽ (il)	n	l	c
k1	m	കൾ (kal)	n	l	c
pof	m	0	n	l	c

In the verbframe file above as example, the first feild gives the name of the verb. SID is the unique sense identification number. It is represented as *verb_root%verb_type%sense number*. The verb_types in Malayalam are distinguished into transtitive, intransitive and causative. Here in the example the type of the verb is transtitive and is represent by *VT*. Verbs_in_Same_Class field gives the list of all the verbs that have same meaning as the given verb. Since Malayalam is a verb final language, by default, all the words are kept normally on the left side of the verb. Rarely, it happens that particular word occurs on the right side of the verb. As an example consider the following sentence:

എങ്കിലും ഭരതനാട്യത്തിന് തന്നെ ശ്രദ്ധ ചെലുത്താൻ ആണ് നീലിമ ഇപ്പോൾ ശ്രമിക്കുന്നത് .

enkiluM bharatanaaTyattinu tanne shRaddha celuttaan aaNu niilima ippoL shRamikkunnatu.

Nilima is now trying to focus on Bharatanatyam itself.

In the above example, "ആണ്" *aaN* is the finite verb and it occurs in between the given sentence. These type of constructions are found mostly. In such cases, the the words that follow the final verbs are positioned on the right side. The verb frame for above example is as follows :

arc_ label	nec- essity	Vibhakti	Lex Type	posn	reln
ccof	m	0	avy	l	c
k4	m	ഉ് (u)	n	l	c
vmod	m	ആൻ (aan)	v	l	c
vmod- emph	m	ഉന്നത് (unnatu)	v	l	c

Verbframe of a similar verb in different sentences varies according to argument relations (Karaka relations) change. For example, for the verb "കാണാം" *kaaNaaM*, different frames are shown below.

Verb::കാണ് *kaaN*
SID::കാണ്%VT%S1
Verb Sense::
Eng_Gloss::to see
Verb Class::
Verb_in_Same_Class::
TAM for the verb root::ആം *aaM*
Frames::

Example::1

വള്ളുവനാടൻ ഗ്രാമങ്ങളും നെൽവയലുകളും ഗത കാല പ്രൗഡിയോടെ നിൽക്കുന്ന മനകളും എങ്ങും കാണാം .

vaLLuvanaaTan graamangngaLuM nelvayalukaLuM gatakaala prauDiyooTe nilkkunna manakaLuM engnguM kaaNaaM.

Valluvanadan villages, paddy fields and the abode houses of Malayali Brahmins with its historical pride can be seen.
FRAME_ID::1

arc_ label	nec- essity	Vibhakti	Lex Type	posn	reln
k1	m	ഉം (uM)	n	l	c
adv	d	0	adv	l	c

Example::2
ഇവിടെ ഉള്ള വ്യൂപോയിന്റിൽ നിന്ന് ഷില്ലോങ്ങ് നഗരത്തിന്റെ , പ്രത്യേകിച്ച് രാത്രിയിൽ ദീപാലംകൃതം ആകുന്ന ഉജ്ജ്വലമായ കാഴ്ച കാണാം .
iviTe uLLa vyuupooyinRil ninnu Silloongng nagarattinRE , pratyeekiccu raatriyil diipaalaMkRItaM aakuna ujjvalamaaya kaazhcha kaaNaM.
From the view point here, the city of Shillong, especially the magnificent view of the city decorated with lights at night can be seen.
FRAME_ID::2

arc_ label	nec- essity	Vibhakti	Lex Type	posn	reln
k5	m	ഇൽനിന്ന് (ilninn)	n	l	c
k1	d	0	n	l	c

Example::3
വിദേശത്ത് ചെന്നാൽ ശ്രീലങ്കയിലും ബാലിയിലും ഒക്കെ രാമയൻ സർക്യൂട്ടുകൾ കാണാം .
videeshattu cennaal sRIlangkayiluM baaliyiluM okke raamayan sarkyuuTTukaL kaaNaaM.
If you go abroad, the Ramayan circuits can be seen especially in Sri Lanka and Bali.
FRAME_ID::3

arc_ label	nec- essity	Vibhakti	Lex Type	posn	reln
vmod	m	0	v	l	c
k7p	m	ഇൽ (il)	n	l	c
k1	d	കൾ (kal)	n	l	c

Example::4
ബക്സറിൽ നിന്ന് അഞ്ച്കിലോമീറ്റർ വടക്കുകിഴക്കുമാറി അഹല്യാദേവിയുടെ അമ്പലം കാണാം .
baksaRil ninnu anjcu kiloomiiRRar vaTakkukizhakku maaRi ahalyaadeeviyuTe ampalaM kaaNaaM.
The temple of Ahalya Devi is located five kilometers north-east from Buxar.
FRAME_ID::4

arc_ label	nec- essity	Vibhakti	Lex Type	posn	reln
k5	m	ഇൽനിന്ന് (ilninn)	n	l	c
vmod	m	ഇ (i)	v	l	c
k1	d	കൾ (kal)	n	l	c

Example::5
ഭാരതത്തിൽ മൂന്നു പ്രധാന വിഭാഗങ്ങളിൽ പെട്ടവരെ കാണാം : ഇന്തോ - ആര്യൻ വംശജർ , ദ്രാവിഡ വംശജർ , മംഗോൾ - ആര്യൻ വംശജർ .
bhaaratattil muunu pradhaana vibhaagangngaLil peTTavare kaaNaaM : intoo-aaryan , draaviDa vaMshajar , maMgooL - aaryan vaMshajar.
There are three main groups in India: Indo-Aryan tribes, Dravidian and Mangol-Aryan tribes.
FRAME_ID::5

arc_ label	nec- essity	Vibhakti	Lex Type	posn	reln
k1	m	അർഎ (are)	v	l	c
k1	d	0	n	r	c

Example::6
ആംഗികം , വാചികം , സാത്വികം , ആഹാര്യം എന്നീ അഭിനയരീതികൾ ഏറ്റക്കുറച്ചിലുകളോടെ മുടിയേറ്റിൽ കാണാം
aaMgikaM, vaacikaM, saatvikaM, aahaaryaM enni abhinjayariitikal eeRRakkuRaccilikaLoTe kaaNaaM.
Different styles of actings like Agikam, Vachikam, Satvikam, Aharyam are found in a ritualistic art form Mudiyettu.
FRAME_ID::6

arc_ label	nec- essity	Vibhakti	Lex Type	posn	reln
k1	d	കൾ *(kal)*	n	l	c
k4	d	കൾഓട്എ (kaLooTe)	n	l	c
k7	m	ഇൽ (il)	n	l	c

It is clear from the above example that verbframe of similar verb is different from the

other verb frames as the argument relations namely the Karaka relations are changing. In the present study, we have taken 3000 dependency annotated sentences for generating the verb frames. Verb frames for 255 verbs [2] were generated from these sentences.

There are some sentences which have 2 finite verbs. Such sentences are not considered in the present study. Examples for such a sentence:
അതിൻറെ അർഥം മനസ്സിലാക്കാൻ ആർക്കും നിയമ സഹായം തേടേണ്ടി വരും എന്നും തോന്നുന്നില്ല .
atinRe arthaM manassilaakkaan aarkkum niyama sahaayaM teeTeeNTi varuM ennuM toonnunnilla.
It does not appear to have any legal assistance to understand the meaning.
In the above sentence, the "എന്നും"(ennuM) is the connector. To this connector, the two finite verbs "വരും"" *varuM* and "തോന്നുന്നില്ല "*toonnunnilla* is joined in the dependency tree. The sentenence is annotated as follows and the dependency tree diagram is depiced in Figure 3.

5 Conclusion and Future Directions

5.1 Conclusion

Application of approach for generating verbframe developed by IIIT-H seems to be adopted for Malayalam languages. However we have to explore how does this can be made applicable for the sentences which have two or more finite verbs.

5.2 Future Directions

This work can be further extended to classify the verb frames according to the semantic nature of the verb. Also attempts can be made to extract the verb frames from dependency annotated corpora though some machine learning approaches.

Acknowledgement

We acknowledge Consortium for Developing Dependency TreeBanks for Indian Languages and its leader IIIT-Hyderabad, especially Prof. Dipti Misra Sharma and Department of Electronics & Information Technology (***DeitY***) , Government of India .

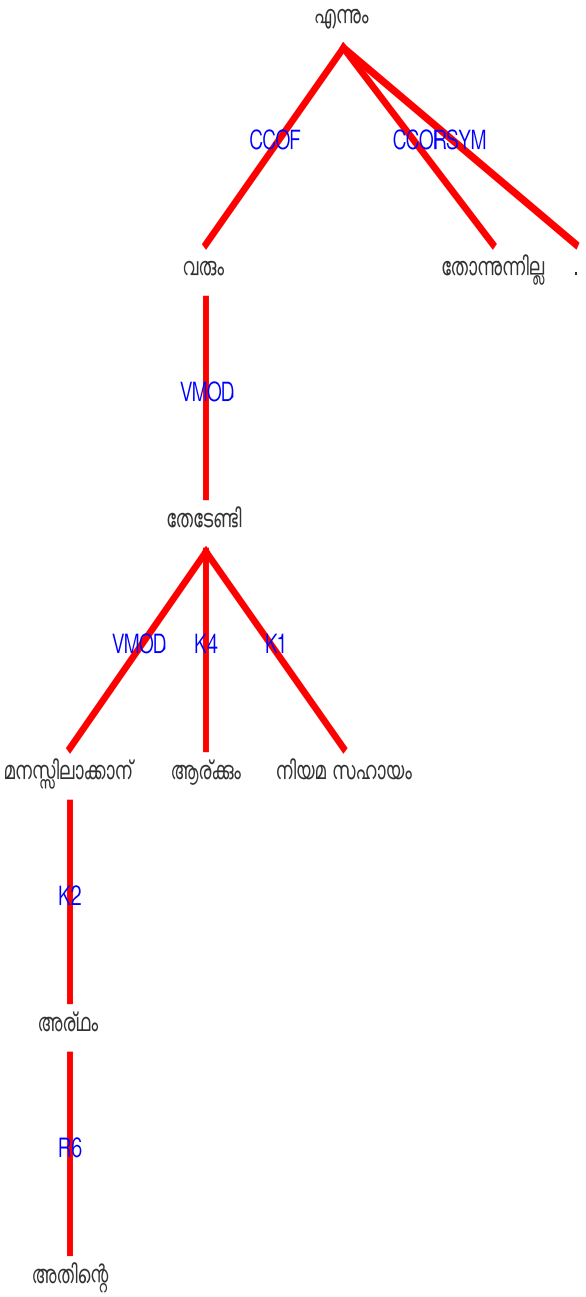


Figure 3: Dependency tree for given example

References

Beth Levin. 1993. *English Verb lasses and Alternations. A Preliminary Investigation*, University of Chicago press.

Beth Levin and Malka Rappaport Hovav. 1995. *Unaccusativity: At the syntax-lexical semantics interface.*, MIT press. 26.

Karin Kipper Schuler. 2005. *VerbNet: A Broad-coverage, Comprehensive Verb Lexicon*, *PhD dissertation, Computer and Information Science Department*, University of Pennsylvania.

[2]Only finite verbs are considered.

Karin Kipper, Hoa Trang Dang and Martha Palmer . 2000. *VerbNet: Class based construction of a verb lexicon*, In *AAAI/IAAI*, 691 696.

Christiane Fellbaum. 1998. *WordNet An Electronic Lexical Database*, MIT press.

Paul Kingsbury and Martha Palmer. 2003. *PropBank: the next level of treebank*, In *Proceedings of Treebanks and lexical Theories*, Vol. 3.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. *The proposition bank: An annotated corpus of semantic roles.*, In *Computational linguistics*, Vol. 31(1) , 71-106.

Paul Kingsbury and Martha Palmer. 2002. *PropBank: the next level of treebank*, In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pp. 1989-1993.

Rafiya Begum, Samar Husain, Lakshmi Bai, and Dipti Misra Sharma. 2008. *Developing Verb Frames for Hindi.*, In *LREC*.

Collin F Baker, Charles J Fillmore, and John B Lowe . 1998. *The Berkeley FrameNet Project*, In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics* , Association for Computational Linguistics. 6:86-90.

Rafiya Begum and Dipti Misra Sharma. 2017. *Development and Analysis of Verb Frame Lexicon for Hindi.*, In *Linguistics and Literature Studies*. 5(1):1-22.

Rafiya Begum, Samar Husain, Arun Dhwaj, Lakshmi Bai, cc and Rajeev Sangal. 2008. *Dependency Annotation Scheme for Indian Languages.*, In *IJCNLP*. 721-726.

Sanjukta Ghosh. 2014. *Making Verb Frames for Bangla Vector Verbs.*, In *Proceedings of 11th Intl. Conference on Natural Language Processing*. 305314.

Ankush Soni, Sambhav Jain, and Dipti Misra Sharma. 2013. *Exploring Verb Frames for Sentence Simplification in Hindi.*, In *IJCNLP*. 1082-1086.

Hanumant Redkar, Sandhya Singh, Nandini Ghag, Jai Paranjape, and Nilesh Joshi. 2016. *Verbframator: Semi-Automatic Verb Frame Annotator Tool with Special Reference to Marathi.*, In *Proceeding s of 13th International Conference on Natural Language Processing*. pp-299.

Schulte im Walde and Sabine. 2009. *The induction of verb frames and verb classes from corpora.*, *Corpus Linguistics. An International Handbook*. Mouton de Gruyter, Berlin.

Akshar Bharati, Vineet Chaitanya, Rajeev Sangal, and K V Ramakrishnamacharyulu. 1995. *Natural language processing: a Paninian perspective.*, Prentice-Hall of India New Delhi.

Joakim Nivre. 2005. *Dependency grammar and dependency parsing.*, In *MSI report*. 5133(1959):1-32.

Akshar Bharati and Rajeev Sangal. 1993. *Parsing free word order languages in the Paninian framework*, In *Proceedings of the 31st annual meeting on Association for Computational Linguistics* Association for Computational Linguistics. 105-111.

Miriam Butt. 2010. *The light verb jungle: still hacking away.*, In *Complex predicates in cross-linguistic perspective* Cambridge University Press. 48-78.

Hindi Shabdamitra: A Wordnet based Tool for Enhancing Teaching-Learning Process

Hanumant Redkar, Nilesh Joshi, Sayali Khare, Lata Popale, Malhar Kulkarni and Pushpak Bhattacharyya

Center for Indian Language Technology,
Indian Institute of Technology Bombay, India.
{hanumantredkar, joshinilesh60, sayali.khare92, popale.lata}@gmail.com,
malharku@gmail.com and pushpakbh@gmail.com

Abstract

Vocabulary building is fundamental to any language learning and effective communication relies on the mastery of vocabulary. Hindi is one of the widely spoken languages in the world. However, there is a scarcity of quality e-learning resources for Hindi. Also, there is lack of e-learning content which is in-sync with Hindi curriculum. This was the motivation in building educational application, *Hindi Shabdamitra*, for language teaching and learning. Hindi Shabhadamitra is an e-learning tool developed using Hindi wordnet for Hindi language learning. This is an insight of the work reported by Redkar et al. (2017) in which Hindi Shabdamitra enhances the teaching-learning process has been presented. The paper presents the teacher and user benefits of this e-learning tool. Further, the user evaluation information has been reported.

1 Introduction

Hindi language is a member of the Indo-Aryan group of the Indo-European language family[1]. Hindi is the official language of India and is 4th among the most spoken languages in the world[2]. Devanagari script is recommended as the official script for Hindi[3]. Hindi in Devanagari script is a part of most of the school curriculum and covers a large spectrum of learners in India, and around the world. A lot of digital content is available online for learning Hindi. Most of this content is in the form of conversation, stories, poems, games, etc. However, less attention is given to learning of the detailed grammatical and lexico-semantic features. Also, there is a scarcity of digital resources in the e-learning domain, especially in the formal setup or curriculum-based setup.

The rapid change in technology and availability of anytime, anywhere digital resources led to the reduction in the cost of delivering the education to the volume of students. As per Shams and Seitz (2008); Sankey et al. (2010), e-learning enhances the performance of the students due to its multi-sensory impact on education. As per Nation and Newton (1997); Lin (1997); Carter (1987), vocabulary teaching and learning should be done in a systematic and structured manner.

Vocabulary learning is considered to be a central activity for any language learning process (Alqahtani et al., 2015). Also, learning grammar is important to understand the syntax and semantics of the word usage. Knowledge of vocabulary is one of the primary reasons for learner's ability and confidence to communicate. Various languages learning strategies like repetition, context, usage and visual correlation were tested by language experts for enhancing the vocabulary (Atasheneh and Naeimi, 2015; Butler et al., 2010).

The multi-modal learning environments have been studied in different settings (Dale, 1969; Mayer and Moreno, 2003; Moreno and Mayer, 2007; Shams and Seitz, 2008; Sankey et al., 2010) which shows its positive impact on learners and always result in better learning and retention. Methods like self-directed technology (Lai et al., 2016), mobile assisted language learning (Yang, 2013) and gamification has arrived as an effective pedagogical strategies. These strategies help the learner to engage and motivate to learn in a relaxed manner (Werbach and Hunter, 2012; Figueroa Flores, 2015). Lexical and semantic relations of words help in better understanding of vocabulary (Lin, 1997).

[1] urlhttps://www.britannica.com/topic/Hindi-language

[2] urlhttps://www.babbel.com/en/magazine/the-10-most-spoken-languages-in-the-world

[3] urlhttp://www.constitution.org/cons/india/p17343.html

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 245–254

The Princeton WordNet[4] or WordNet, have been explored by Hu et al. (1998); Sun et al. (2011); Brumbaugh (2015); Hiray (2015) for vocabulary learning and other related language learning applications.

Keeping vocabulary and grammar learning as pivotal to language learning, an e-learning tool, *Hindi Shabdamitra*, has been developed based on Hindi Wordnet[5] (Redkar et al., 2017). This paper presents an insight of the features and enhancements in the teaching-learning process using Hindi Shabdamitra.

The rest of the paper is organized as follows - section 2 presents Hindi Shabdamitra, section 3 explains the functional and unique features of Hindi Shabdamitra and the comparative study, section 4 provides enhancements in teaching-learning and the benefits of end-users, section 5 describes the user evaluation and field response. Finally, in section 6 the paper is concluded with the scope for future work.

2 Hindi Shabdamitra

Hindi Shabdamitra (हिंदी शब्दमित्र, *hiMdI shabdamitra*)[6] is a digital aid designed for assisting in teaching and learning Hindi language (Redkar et al., 2017). The end users of this e-learning tool are students, teachers, parents and self-learners. It is developed for formal teaching-learning environment as well as informal learning environment. The formal setup is designed in correlation with school curriculum. Self-learners, organizations, NGOs, NRIs, *etc.*, belong to the informal learning environment. It uses Hindi Wordnet as a base resource that has been remodeled for this aid by incorporating the multi-modal features. Further, the concepts are grammatically enriched and simplified depending upon the understanding level of the learner.

A team of lexicographers, illustrators and native language speakers has contributed to build this multi-modal resource which has formed the base of Hindi Shabdamitra. The tool has an online web-based and app-based interface for wider reachability. Also, this tool can be made available offline for anytime, anywhere learning. The interface allows the search navigation in two ways – level wise (हिंदी ज्ञान स्तर के अनुसार, *hiMdii GYaana stara ke anusaara*) and class wise (कक्षा के अनुसार, *kaxaa ke anusaara*). This can be seen in figure 1 and 2.

Figure 1: Features of a word पात (*pAta*, leaf) for Class 8, lesson 1 in class-wise search interface

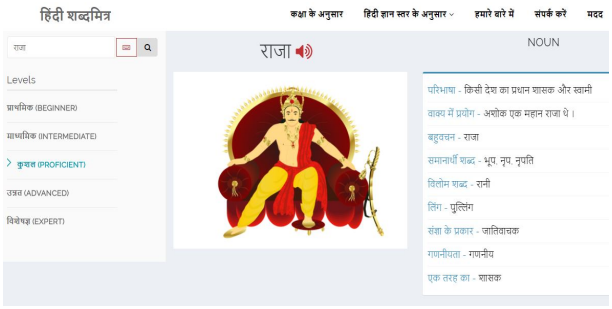


Figure 2: Features of a word राजा (*rAjA*, a king) for proficient level in level-wise search interface

Layered Interface

Hindi Shabdamitra interface has a layered architecture. Depending on the proficiency of the learner, the layers have been designed for learners, scaling from the beginner to the proficient. The level of a leaner is determined by using the class-wise proficiency selection criteria as shown in table 1.

The same word can be studied by the learners at all 5 levels. At each level, incremental information is displayed. The depth of content displayed vary from level to level. The detailed layer-wise list of features have been provided in the next section.

Layer	Level	Class
Layer 1	Beginner (प्राथमिक)	1 and 2
Layer 2	Intermediate (माध्यमिक)	3, 4 and 5
Layer 3	Proficient (कुशल)	6, 7, 8, 9 and 10
Layer 4	Advanced (उन्नत)	11 and 12
Layer 5	Proficient (विशेषज्ञ)	Above 12

Table 1: Proficiency selection criteria for Hindi Shabdamitra

[4] https://wordnet.princeton.edu/

[5] http://www.cfilt.iitb.ac.in/wordnet/webhwn/

[6] http://www.cfilt.iitb.ac.in/hindishabdamitra/

3 Features of Hindi Shabdamitra

Hindi Shabdamitra provides functional features and unique features for the learners. These features help learners in clearly understanding the concept. Also, it aids the teachers and parents to explain the concept in detail. The list of features with respect to the classes and levels is given in table 2. Following are the detailed description of functional and unique features of Hindi Shabdamitra. These features are explained with the help of Matthews (2007), Oxford dictionaries[7], Online dictionary for language technology[8], Cambridge dictionary[9], Hindi Wordnet[10], *etc.*

Functional Features

The functional features are most commonly found in regular dictionaries or thesauri. These features are available for all the levels.

1. **Part of Speech (POS)**: Hindi Shabdamitra has content based on the part of speech *viz.*, noun, adjective, verb and adverb, as it is based on the original resource, Hindi Wordnet.

    - **Noun / संज्ञा (*saMGYaa*)** : One of the classes of words whose characteristic role is as an argument of a verb and which is characteristically that of words denoting concrete entities (Matthews, 2007). Nouns are known as संज्ञा (*saMGYaa*) in Hindi. Just like nouns in English, संज्ञा (*saMGYaa*) is also a names of people, places, things and ideas. For example:
        - कमला (*kamalaa*, name of a lady)
        - दिल्ली (*dillii*, the capital city of India)
        - मिठास (*miThaasa*, sweetness)
        - गणित (*gaNita*, mathematics)

    - **Adjective / विशेषण (*visheShaNa*)** : A word of class whose most characteristic role is as the modifier of a noun. Adjectives are known as विशेषण (*visheShaNa*) in Hindi. For example:
        - चतुर (*chatura*, clever)
        - काला (*kaalaa* , black)
        - लम्बा (*lambaa*, tall)
        - छोटा (*ChoTaa*, small, young)

    - **Verb / क्रिया (*kriyaa*)** : One of a class of lexical units whose characteristic syntactic role is as a predicate or predicator and which is characteristically that of words denoting actions or processes. The verb, specifically the action verb, is known as क्रिया (*kriyaa*) in the Hindi language. For example:
        - खाता है (*khaataa hai*, eats)
        - पीता है (*piitaa hai*, drinks)
        - जाता है (*jaataa hai*, goes)
        - खेलता है (khelataa, plays)

      Note that the verbs given in the example are for singular masculine subjects. For feminine gender and singular number, the verbs are conjugated as खाती है (*khAtI hai*), पीती है (*pItI hai*) and जाती है (*jAtI hai*). The plurals of the verbs for both the genders are खाते हैं (*khAte haiM*), जाते हैं (*jAte haiM*) and पीते हैं (*pIte haiM*).

	Levels	1	2	3	4	5
	Classes	**1, 2**	**3, 4, 5**	**6 to 10**	**11, 12**	**12+**
#	Features					
1	POS	Yes	Yes	Yes	Yes	Yes
2	Multiple Senses/Polysemy	Yes	Yes	Yes	Yes	Yes
3	Audio Pronunciation	Yes	Yes	Yes	Yes	Yes
4	Illustration/Picture	Yes	Yes	No	No	No
5 (a)	Simplified gloss & example	Yes	Yes	No	No	No
5 (b)	Original gloss & example	No	No	Yes	Yes	Yes
6	Gender	No	Yes	Yes	Yes	Yes
7	Synonym	Yes	Yes	Yes	Yes	Yes
8	Antonym	Yes	Yes	Yes	Yes	Yes
9	Number	No	Yes	Yes	Yes	Yes
10	Countability	No	Yes	Yes	Yes	Yes
11	Affix	No	No	Yes	Yes	Yes
12	Junction	No	No	Yes	Yes	Yes
13	Kinds of POS	No	No	Yes	Yes	Yes
14	Indeclinable	No	No	Yes	Yes	Yes
15	Spelling Variation	No	Yes	Yes	Yes	Yes
16	Transitivity	No	No	Yes	Yes	Yes
17	Hypernymy/Is Kind Of	No	No	Yes	Yes	Yes
18	Hyponymy/Type Of	No	No	Yes	Yes	Yes
19	Meronymy	No	No	No	Yes	Yes
20	Holonymy	No	No	No	Yes	Yes
21	Modifies Verb	No	No	No	Yes	Yes
22	Modifies Noun	No	No	No	Yes	Yes
23	Troponymy	No	No	No	No	Yes
24	Causative	No	No	No	No	Yes
24	Entailment	No	No	No	No	Yes
25	Link Type	No	No	No	No	Yes
26	Attribute	No	No	No	No	Yes

Table 2: Class-wise and level-wise features of Hindi Shabdamitra

[7] https://en.oxforddictionaries.com

[8] http://www.odlt.org/

[9] http://dictionary.cambridge.org/

[10] http://www.cfilt.iitb.ac.in/wordnet/webhwn/index.php

- **Adverb /** क्रियाविशेषण **(*kriyaavisheSha-Na*)** : A word of class whose most characteristic role is traditionally that of modifying a verb or verb phrase. For example:
  - धीरे (*dhiire*, slowly)
  - जल्दी (*jaldii*, fast)
  - पास) (*paasa*, near)
  - आज) (*aaja*, today)

2. **Multiple Senses/Polysemy** : Polysemy refers to the situation where the same word has two or more different meanings. In level-wise interface of Hindi Shabdamitra polysemy is provided where a user can click on the next button to get the another meaning of same word, if available. However, in class-wise interface there is only one word meaning, as it corresponds to the meaning with the textbooks.

3. **Audio pronunciation** : The audio pronunciations are recorded by the native Hindi speakers, as to promote 'Standard Hindi'. Hindi Shabdamitra is provided with the audio of the search-word.

4. **Illustration (Image / Picture)** : Illustrations are drawn, keeping in the mind Indian context. Illustrations are targeted to the learners of level 1 and level 2 of the Hindi Shabdamitra. Following are the things taken into consideration while illustrating a concept:
   - Omitting the sensitive context, *e.g.*, शव (*shava*, a dead body). As it may cause adverse effect on the kid, such illustrations are omitted or expressed in a different way.
   - Overlapping: If an illustration is drawn for पूजा करना (*pUjA karanA*, to worship), it overlaps with आरती करना (*AratI karanA*, do ritual) as well. Such instances are avoided.
   - One illustration for multiple concepts: The illustration for बड़ा (*baDaA, big*) has its counterpart *i.e.*, छोटा (*ChoTA*, small). Here, for these concepts single illustration can be used.
   - Abstract nouns: Abstract nouns are illustrated with the base of provided example or a specific situation. For *e.g.*, साथ (*saatha*, company). A sample illustration of an abstract noun साथ is shown in figure 3.

Figure 3: Illustration of a word साथ (*saatha*)

.

5. **Gloss or Definition** :
   The concept is defined by providing gloss or definition of a word. Gloss is usually picked from Hindi wordnet, however, it is simplified in special cases.

   (a) **Simplified Gloss and Example**: Hindi Shabdamitra has provided simplified gloss for level 1 and 2, *i.e.* for class 1 to 5. The gloss and example sentence from original Hindi Wordnet is further simplified by providing the simple words in the definition so that the students at these levels can easily pick-up and learn concept comfortably.
   If Hindi Wordnet gloss is completely or partly difficult to understand then it is simplified. For example, for a concept नदी (*nadI*, river) having IndoWordNet[11] Synset id 4430, the original Hindi Wordnet gloss is जल का वह प्राकृतिक प्रवाह जो किसी पर्वत से निकलकर निश्चित मार्ग से होता हुआ समुद्र या किसी दूसरे बड़े जल प्रवाह में गिरता है (*jala kA vaha prAkRRitika pravAha jo kisI parvata se nikalakara nishchita mArga se hotA huA samudra yA kisI dUsare baDae jala pravAha meM giratA haiM*, a large natural stream of water) (Bhattacharyya, 2010).

[11] http://www.cfilt.iitb.ac.in/indowordnet/

This concept is difficult to understand by a level 1 and 2 learners, hence this gloss is simplified as पर्वत से निकलकर अपने आप बहती हुई पानी की धारा (*parvata se nikalakara apane Apa bahatI huI pAnI kI dhArA*, river). Similarly, original Hindi Wordnet example sentence, गंगा, यमुना, सरस्वती, सतलुज, कावेरी, सरयू आदि भारत की प्रमुख नदियाँ हैं। (*gaMgA, yamunA, sarasvatI, sataluja, kAverI, sarayU Adi bhArata kI pramukha nadiyAMN haiM*, Ganga, Yamuna, Saraswati, Satluj, Kaveri, Sharayu, *etc.* are India's major rivers) is simplified to "गंगा, यमुना बड़ी नदियाँ हैं। (*gaMgA, yamunA baDaI nadiyAMN haiM*, Ganga, Yamuna are major rivers).

(b) **Original Hindi Gloss** : If Hindi Wordnet gloss is easy to understand, it was kept as it is for level 1 and 2. For *e.g.*, for a concept पैदल (synset Id: 6274), Hindi Wordnet gloss is पैरों से चलकर (*pairoM se chalakara*, afoot) and Hindi Wordnet example is वह विद्यालय पैदल जाता है। (*vaha vidyAlaya paidala jAtA hai*, he is going to the school afoot). For levels 3, 4 and 5, the original gloss is kept as it is, as at these levels, learners will understand the concepts by reading the original Hindi Wordnet gloss and examples, hence it is not simplified further.

6. **Gender /** लिंग **(*liMga*)** : Grammatical category dividing nouns into classes basically characterizable by reference to sex. In Hindi, there are two genders:
   - **Masculine /** पुल्लिंग **(*pulliMga*)**: nouns denoting males.
   - **Feminine /** स्त्रीलिंग **(*strIliMga*)**: nouns denoting females.

7. **Synonym /** समानार्थी शब्द **(*samAnArthI shabda*)** : Two lexical units with a shared meaning. A synonym is a word or phrase that means exactly or nearly the same as another word or phrase in the same language. Words that are synonyms are said to be synonymous, and the state of being a synonym is called synonymy. For example, मित्र (*mitra*, friend) and दोस्त (*dosta*, friend) are synonyms.

8. **Antonym /** विलोम शब्द **(viloma shabda)** : Words that have opposite meaning. In lexical semantics, opposites are words lying in an inherently incompatible binary relationship, like the opposite pairs बड़ा (*badaa*, big) - छोटा (*ChoTaa*, small).

9. **Number /** वचन **(*vachana*)** : Inflectional category basically distinguishing reference to one individual from reference to more than one.
   - **Singular /** एकवचन **(*ekavachana*)**: Term in the category of number, at least one of whose roles is in referring to one individual as opposed to more than one.
   - **Plural /** बहुवचन **(*bahuvachana*)**: Term in the category of number, referring to more than one, or more than some small number of individuals.

10. **Countability /** गणनीयता **(*gaNanIyatA*)** :
    - **Countable /** गणनीय **(*gaNanIya*)**: A noun whose syntax is that of ones denoting individuals that can be counted.
    - **Uncountable /** अगणनीय **(*agaNanIya*)**: A noun whose syntax is characteristic of a class whose members do not denote individuals that can be counted.

Unique Features

These features provide the deeper grammatical information, *i.e.*, lexico-semantic relations. These are not commonly found in regular dictionaries or thesauri. These features are made available from level 3.

11. **Affix** : Any element in the morphological structure of a word other than a root.
    - **Prefix /** उपसर्ग **(*upasarga*)**: An affix which comes before the form to which it is joined.
    - **Suffix /** प्रत्यय **(*pratyaya*)**: An affix that comes after the form to which it is added.
    - **Root word /** मूल शब्द **(*mUla shabda*)**: A form from which words or parts of words are derived and which is not itself derivable from any smaller or simpler form.

12. **Junction /** संधि **(*saMdhi*)** : Ancient Indian term for the modification and fusion of sounds at or across the boundaries of grammatical units. The major are स्वर (*svara*, vowel), व्यंजन (*vyaMjana*, consonant) and विसर्ग (*visarga*).

13. **Kind of POS**

Kind of Noun / संज्ञा के प्रकार **(*saMjñA ke prakAra*)** :

- **Proper noun /** व्यक्तिवाचक संज्ञा **(*vyaktivAchaka saMjñA*)** : It is a name that identifies a particular person, place, or thing. These nouns refer to something particular. For example, writeHi (*govA*, Goa) and शिवाजी (*shivAjI*, Shivaji) are proper nouns.

- **Common noun /** जातिवाचक संज्ञा **(*jAtivAchaka saMjñA*)** : Common Nouns are words used to describe similar items or a class of things. For example, a person who studies is a student, a person who practices medicine as a profession is a doctor. किताबें (*kitAbeM*, books), आदमी (*AdamI*, man), दोस्त (*dosta*, friend), and लड़की (*laDakI*, girl) are common nouns.

- **Abstract Noun /** भाववाचक संज्ञा **(*bhAvavAchaka saMjñA*)** : It is a noun which refers to ideas, qualities, and conditions - things that cannot be seen or touched and things which have no physical reality. सत्य (*satya*, truth), बहादुरी (*bahAdurI*, bravery), खुशी (*khushI*, happiness) are examples of abstract nouns.

- **Collective Noun /** समूहवाचक संज्ञा **(*samUhavAchaka saMjñA*)** : It refers to groups of people or things. Names are given to a collection of persons or things are known as समूहवाचक संज्ञा (*samUhavAchaka saMjñA*). These words are used to refer to the whole group as one. Examples are सेना (*senA*, army), सभा (*sabhA*, assembly), परिवार (*parivaar*, family) are examples of collective nouns in Hindi.

Kind of Adjective / विशेषण के प्रकार **(*visheShaNa ke prakAra*)** :

- **Qualitative /** गुणवाचक **(*guNavAchaka*)** : Describes the qualities of a person or thing. Examples are अच्छा(*achChA*, good, nice), लम्बा (*lambA*, Long), बाहरी (*bAharI*, outsider), लाल (*lAla*, red), *etc.*

- **Numeral /** संख्यावाचक **(*saMkhyAvAchaka*)** : Denotes number of person or things. Examples are दो (*do*, two), कई (*kaI*, many), थोड़े (*thoDae*, few), *etc.*

- **Quantitative /** परिमाणवाचक **(*parimANavAchaka*)** : denotes the amount or quantity. Examples are दो किलो (*do kilo*, two kilograms), ज्यादा (*jyaadA*, Plenty, More), *etc.*

- **Pronominal /** सार्वनामिक **(*sArvanAmika*)** : A pronoun that is being used as an adjective to modify a noun or another pronoun. Examples are मेरी पुस्तक (*merI pustaka*, my book), किसका काम (*kisakA kAma*, whose work), आपका सामान (*ApakA sAmAna*, your stuff).

Kind of Verb / क्रिया के प्रकार **(*kriyaa ke prakaara*)** :

- **Simple verb /** सरल क्रिया **(*sarala kriyaa*)** : In Hindi, there are some verbs that are composed of single word. These verbs are called simple verbs. For example, पूछना (*puuChanaa*, to ask), होना (*honaa*, to be), *etc.*

- **Conjunct verb /** संयुक्त क्रिया **(*saMyukta kriyaa*)**: It has different combinations of more than one POS categories, *viz.*, Noun + Verb, Adjective + Verb. A conjunct verb is formed by joining either a noun or an adjective with a verb. For example, वह साफ़ करती है (*vaha saapha karatii hai*, she cleans).

- **Compound verb /** यौगिक क्रिया **(*yaugika kriyaa*)**: It has Verb + Verb combination, where one verb is semantically contented and the other verb acts as a modifier. For example, ले जाना (*le jaanaa*, to carry away), *etc.*

- **Causative verb /** प्रेरणार्थक क्रिया **(*preraNaarthaka kriyaa*)**: Causative verb denotes an action which is not directly performed by the subject but indirectly through some other agent. These are verbs which end in "वाना" (*vaanaa*) and

"आना" (*aanaa*). Example are, करवाना (*karavaanaa*, to cause to do) and कराना (*karaanaa*, to cause to do).

Kind of Adverb / क्रिया विशेषण के प्रकार **(*kriyA visheShaNa ke prakAra*)** :

- **Manner /** रीतिवाचक **(*rItivAchaka*)** : Manner adverbs tell us about the way something happens or is done. For example:ध्यानपूर्वक (*dhyAnapUrvaka*, carefully).
- **Place /** स्थानवाचक **(*sthAnavAchaka*)** : Place adverbs tell us about where something happens or where something is. For example: ऊपर (*Upara*, above).
- **Time /** कालवाचक **(*kAlavAchaka*)** : Time adverbs tell us about when something happens. For Example : अभी (*abhI*, now).
- **Quantity /** परिमाणवाचक **(*parimANavAchaka*)** : Quantity adverbs modify the quantity or intensity of an adjective or verb. For example : बिल्कुल (*bilkula*, surely, totally).

14. **Indeclinable /** अव्यय **(*avyaya*)** : A form which does not have a distinct inflections. For example पश्चात (*paSHchaat*, subsequently).

15. **Spelling variation /** शब्द–विन्यास विविधता **(*shabda-vinyAsa vividhatA*)** : Words which are having variation in their spellings. For example नज़र (*najara*, vision) and नजर (*najara*, vision) has spelling variations.

16. **Transitivity /** संक्रामिता **(*saMkrAmitA*)** :

    - **Transitive /** सकर्मक **(*sakarmaka*)** : Construction in which a verb is related to at least two nouns or their equivalent, whose semantic roles are characteristically those of an agent and a patient. For example खाना (khaanaa, to eat).
    - **Intransitive /** अकर्मक **(*akarmaka*)** : A construction in which a verb is related to a single noun or its equivalent. For example सोना (sonaa, to sleep).

17. **Hypernymy (is a kind of) /** एक तरह का **(*eka taraha kA*)** : A semantic relation between two synsets to capture super-set hood. For example, hypernymy of आम (*aama*, mango) is फल (*phala*, fruit).

18. **Hyponymy (type of) /** प्रकार **(*prakAra*)** : A semantic relation between two synsets to capture sub-set hood. For example, hyponymy of आम (*aama*, mango) is दशहरी (*dashaharii*, Dashahari).

19. **Meronymy (part of) /** का हिस्सा **(*kA hissA*)** : Relation between lexical units where the objects, *etc.*, denoted by one are parts of those denoted by other. For example, meronymy of आम (*aama*, mango) is गुठली (*guThalii*, seed).

20. **Holonymy (whole of) /** अंगिवाची **(*aMgivAchI*)** : A semantic relation that holds between a whole and its parts. For example, Holonymy of गुठली (*guThalii*, seed) is आम (*aama*, mango).

21. **Modifies Verb /** अर्थ–संकुचन क्रिया **(*arthasaMkuchana kriyaa*)** : Certain adverbs can only go with certain verbs. Modifies Verb is a relation to show connection between such words. For example बाद (*baad*, beyond) modifies verbs होना (*honaa*, to be) and काम करना (*kaam karanaa* , to work).

22. **Modifies Noun /** अर्थ–संकुचन संज्ञा **(*arthasaMkuchana saMGYaa*)** : Certain adjectives can only modify certain nouns. For example अधिक (*adhika*, much) modifies nouns वस्तु (*vastu*, thing), जीव (*jiiva*, being), *etc*.

23. **Troponymy /** प्रकारवाची **(*prakaaravaachii*)** : Troponym denotes a specific manner elaboration of another verb. It shows manner of an action, *i.e.*, X is a troponym of Y if to X is to Y in some manner. For example, मुस्कुराना (*muskuraanaa*, to smile) is a troponym of हँसना (*hansnaa*, to laugh).

24. **Causative /** प्रेरणार्थक क्रिया **(*preraNaarthaka kriyaa*)** : The Causative relation links the causative verbs with the base verbs and show interdependency between them. For example, खाना (*khaanaa*, to eat) has causative verb खिलाना (*khilaanaa*, to make someone eat).

25. **Entailment /** अपरिहार्यतावाची **(*aparihaaryataavaachii*)** : Entailment refers to a relationship between two verbs. Any verb A entails B, if the truth of B follows logically from the truth of A. The relation of entailment is unilateral, *i.e.*, it is one way relation. For example, खर्राटा लेना (*kharraaTaa lenaa*, to snore) entails सोना (*sonaa*, to sleep).

26. **Link Type** : This has major three links as followed:

    - **Ability Link** आन्तर–योग्यता निर्देशी क्रिया **(*aantara-yogyataa nirdeshii kriyaa*)** : This link specifies the inherited features of a nominal concept. For example मछली (*maCHlii*, fish) has ability link to तैरना (*tairnaa*, to swim).
    - **Capability Link** बाह्य–योग्यता निर्देशी क्रिया **(*baahya-yogyataa nirdeshii kriyaa*)** : This link specifies the acquired features of a nominal concept. For example व्यक्ति (*vyakti*, person) has capability link to तैरना (*tairnaa*, to swim).
    - **Function Link** क्रम निर्देशी क्रिया **(*krama nirdeshii kriyaa*)** : This link specifies the function of a nominal concept. For example शिक्षक (*shikshak*, teacher) has functional link to पढ़ाना (*paRhaanaa*, to teach).

27. **Attribute** गुणवाची **(*guNavaachii*)** : This denotes the properties of noun. It is a linkage between noun and an adjective. For example, पक्षी (*pakshii*, bird) has an attribute पंखदार (*pankhdaar*, having wings).

3.1 A Comparative Study

A study of current digital resources used by various educational institution was done as a part of the background study. The outcome showed a big gap of quality resources which can cover aspects of language learning *viz.* grammar, concepts, usage and pronunciations in an effective manner; and which are based on the curriculum.

Some of the applications for language learning which offer Hindi learning are Duolingo[12], Hindipod[13], Rocket Language[14], Italki[15], *etc*. Some applications made specifically for children are dinolingo[16] , akhlesh[17], galligallisimsim[18], *etc*. Other online resources for Hindi language learning are bilingual dictionaries which provide only the meanings of the words, such as Shabdkosh[19], Collins dictionary[20], *etc*.

The common factor among all the above resources is their inability of customization for formal school setups. They are more focused on individual learning (Redkar et al., 2017).

Advantages of Hindi Shabdamitra over the above e-learning tools:

- Hindi Shabdamitra provides an insight about deep grammatical features of a word/concept.
- It caters to school teachers, students and parents by providing the curriculum based vocabulary.
- It is based on a lexically rich resource, Hindi Wordnet, whose features like lexico-semantic and ontological relations provide much more information.
- It provides both, systematic learning as well as random learning approaches.

4 Enhancing Teaching-Learning Process

According to (Dike, 1989), audio-visual resources do not only increase the motivation of the teachers and learners, they add clarity to the topic taught and make learning more interesting. The impact of new technologies in educational contexts has been mostly positive as new technologies have given educators the opportunity to enhance their knowledge, skills, and therefore enhance the standard of education. Researchers have found that student engagement, achievement and motivation are enhanced through integration of such technologies.

Hindi Shabdamitra facilitates learning with the help of illustrations and pronunciation for multisensory impact. This tool can assist the teachers in better classroom management and make learning Hindi an interesting activity (Redkar et al., 2017).

Learning Benefits

With the help of audio-visual methods for learning, small children learn easily and effortlessly. Hence, for the initial phases, in Hindi Shabdamitra, concepts are pictorially depicted by providing illustrations for level 1 and 2, as to understand a concept easily. Most of the illustrations are simple and convey the exact information.

Also, the words are provided with the audio pronunciation. These words are recorded by native

[12] https://www.duolingo.com/
[13] https://www.hindipod101.com/
[14] https://www.rocketlanguages.com
[15] https://www.italki.com
[16] https://dinolingo.com
[17] http://www.akhlesh.com/
[18] http://www.galligallisimsim.com/
[19] www.shabdkosh.com/
[20] https://www.collinsdictionary.com

speakers of the language. The best thing about having audio pronunciation is that the teacher can play it multiple times, until the student understands it properly. Another advantage is that the audio reaches to the entire classroom, *i.e.,* till the last bench.

Hindi Shabdamitra promotes following aspects with the help of audio-visual aids for learning:

- **Experiential Learning:**
  Digital learning redefines the boundaries of a classroom. Sitting in a class, the students experience the application of a concept through dynamic content [21]

- **Flipped Classroom approach:**
  The reversal of traditional teaching which provides active learning. Students are accustomed to interacting with audio and video on electronic devices, so it stands to reason that they would digest educational content in this manner as well[22].

Teaching benefits

- **Validation**:
  Words and their respective illustrations and audio pronunciations give an advantage to the teacher to validate whether the students understood the entire concept or not. S/he is able to present the image and audio multiple times for the clarity of the concept.

- **Sharing the burden**:
  It is certainly difficult for a teacher to provide the meaning of every word. Since Hindi Shabdamitra is focused on vocabulary learning, the teacher has an aid. Whenever a student has a doubt about a specific word s/he can simply search the word. The tool will share the burden of the teacher.

- **Effectiveness as an aid**:
  The teaching profession is filled with countless opportunities to enrich the academic lives of students, while some concepts and educational objectives will be easy for students to grasp, other will require you to think creatively to ensure that important learning objectives are met. Using audiovisual aids in teaching is one way to enhance lesson plans and give students additional ways to process subject information (Kunari, 2006).

[21] https://www.academia.edu/33866288/DIGITAL_CLASSROOMS_A_BOON_FOR_ACHIEVING_QUALITY_EDUCATION_IN_INDIA

[22] https://www.academia.edu/16071067/Effects_of_Flipped_Classroom

5 Field Trial and User Response

As a part of testing the tool, the field trial of Hindi Shabdamitra interface was done at some schools with students & teachers' participation in the exercise. Also, an online survey was conducted. The feedback was sought for the content, ease of handling the application, classroom impact and overall experience by teachers and students. Following are the summarized observations:

- Hindi Shabdamitra helped teachers in explaining concepts clearly with the help of illustrations and simplified concepts.

- The aid assisted the teacher in better classroom management

- Reduced effort of reiterating the concepts for better retention.

- Audio clips helped in understanding the pronunciation of a given word.

- Having the standardized pronunciation by the native Hindi speaker.

The application has been improved based on the feedback received by students and teachers.

6 Conclusion and Future Work

The paper presents an insight of a Hindi Shabdamitra, an e-learning tool for teaching and learning Hindi language by Redkar et al. (2017). A comparative study of the existing technologies with Hindi Shabdamitra was done where it is found to have much more features. The field trial and survey was conducted with which the observations and student & teacher benefits have been reported here. This tool definitely enhances the teaching-learning experience and has an exhaustive feature list which is not there in many of the other e-learning products. It caters to a wide range of audience and is available in both web-based and app-based formats.

In future, the authors intend to record the learning process of student, provide a teaching-learning process flow, an interactive assessment module for evaluations and other game based assessment modules for fun learning.

Acknowledgements

The authors would like to acknowledge the support and help by the members of Center for Indian Language Technology (CFILT)[23] and *Hindi Shabdamitra* team. The funding agency, Tata Center for Technology and Design (TCTD)[24] has been instrumental and supportive throughout the development of *Hindi Shabdamitra*.

References

Mofareh Alqahtani et al. 2015. The importance of vocabulary in language learning and how to be taught. *International Journal of Teaching and Education* 3(3):21–34.

Nasser Atasheneh and Maki Naeimi. 2015. Vocabulary learning through using mechanical techniques vocabulary learning strategy. *Theory and Practice in Language Studies* 5(3):541.

Pushpak Bhattacharyya. 2010. Indowordnet. In *The WordNet in Indian Languages*, Springer, pages 1–18.

Heidi Brumbaugh. 2015. *Self-assigned ranking of L2 vocabulary: using the Bricklayer computer game to assess depth of word knowledge*. Ph.D. thesis, Arts & Social Sciences:.

Shari Butler, Kelsi Urrutia, Anneta Buenger, Nina Gonzalez, M Hunt, and Corinne Eisenhart. 2010. A review of the current research on vocabulary instruction. *National Reading Technical Assistance Center, RMC Research Corporation* 1.

Ronald Carter. 1987. Vocabulary and second/foreign language teaching. *Language Teaching* 20(01):3–16.

Edgar Dale. 1969. Audiovisual methods in teaching .

H L Dike. 1989. Strategies for producing instructional materials .

Jorge Francisco Figueroa Flores. 2015. Using gamification to enhance second language learning. *Digital Education Review* 27:32–54.

Amit C. Hiray. 2015. *Teaching and Learning of EAP Vocabulary: A Web-based Integrative Approach at the Tertiary Level in India*. Ph.D. thesis, Dept. of HSS, IIT Bombay.

X Hu, AC Graesser, Tutoring Research Group, et al. 1998. Using wordnet and latent semantic analysis to evaluate the conversational contributions of learners in the tutorial dialog. In *Proceedings of the international conference on computers in education*. volume 2, pages 337–341.

[23] http://www.cfilt.iitb.ac.in

[24] http://www.tatacentre.iitb.ac.in/digital_aid.php

Kunari. 2006. Methods of teaching educational technology. *Science Direct, New Delhi* .

Chun Lai, Mark Shum, and Yan Tian. 2016. Enhancing learners' self-directed use of technology for language learning: the effectiveness of an online training platform. *Computer Assisted Language Learning* 29(1):40–60.

Chih-Cheng Lin. 1997. Semantic network for vocabulary teaching. *Journal of National Taiwan Normal University* (42):43–54.

P. H. Matthews. 2007. *The Concise Oxford Dictionary of Linguistics*, volume 2nd Edition. Oxford University Press.

Richard E Mayer and Roxana Moreno. 2003. Nine ways to reduce cognitive load in multimedia learning. *Educational psychologist* 38(1):43–52.

Roxana Moreno and Richard Mayer. 2007. Interactive multimodal learning environments. *Educational psychology review* 19(3):309–326.

Paul Nation and Jonathan Newton. 1997. Teaching vocabulary. *Second language vocabulary acquisition* pages 238–254.

Hanumant Redkar, Sandhya Singh, Meenakshi Somasundaram, Dhara Gorasia, Malhar Kulkarni, and Pushpak Bhattacharyya. 2017. Hindi shabdamitra: A wordnet based e-learning tool for language learning and teaching pages 23–28.

Michael Sankey, Dawn Birch, and Michael Gardiner. 2010. Engaging students through multimodal learning environments: The journey continues. In *Proceedings ASCILITE 2010: 27th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education: Curriculum, Technology and Transformation for an Unknown Future*. University of Queensland, pages 852–863.

Ladan Shams and Aaron R Seitz. 2008. Benefits of multisensory learning. *Trends in cognitive sciences* 12(11):411–417.

Koun-Tem Sun, Huang Yueh-Min, and Liu Ming-Chi. 2011. A wordnet-based near-synonyms and similar-looking word learning system. *Journal of Educational Technology & Society* 14(1):121.

Kevin Werbach and Dan Hunter. 2012. *For the win: How game thinking can revolutionize your business*. Wharton Digital Press.

Jaeseok Yang. 2013. Mobile assisted language learning: review of the recent applications of emerging mobile technologies. *English Language Teaching* 6(7):19–25.

"A pessimist sees the difficulty in every opportunity; an optimist sees the opportunity in every difficulty" – Understanding the psycho-sociological influences to it

Upendra Kumar, Vishal Singh Rana, Srinivas PYKL and Amitava Das

Indian Institute of Information Technology, Sri City, AP, India, 517588

{upendra.k14, vishal.s14, srinivas.p amitava.das}@iiits.in

Abstract

This paper presents an empirical study to understand *how psycho-sociological factors influence on optimism/pessimism at the individual level*. Optimists believe that future events are going to work out for the best; pessimists expect the worst. Their expectations manifest in their day-to-day behaviour and also reflects the way a person tweets or behaves in online social media. To this end, we have identified optimist/pessimist users from Twitter, analyzed their personality (psychological) and values & ethics (sociological) at the community level. Empirical analysis reveals some interesting insights and behavioral patterns related to user level optimism/pessimism in different combinations of psychological and sociological factors, are reported.

1 Introduction

Optimists are people who tend to have favourable outlook of their life whereas pessimists tend to derive negative interpretations from the events around them. Their approach towards life generally manifests in their day-to-day behaviour. For example, they have different mechanisms to cope with positive or negative events around them. The first computational model to estimate the degree of optimism/pessimism at tweet level as well as user level has been introduced by [Ruan et al.2016]. This paper could be seen as an extension to that work. Here we are trying to understand how psycho-sociological factors influence on optimism/pessimism at individual level. To understand psychological factors we have analyzed personality of each user using the *Big5 Personality Model* [Goldberg1993]. On the other hand, to understand sociological factors we have used values & ethics (*Schwartz Values Model*) [Schwartz1992] model. Therefore, both the influencing factors have been aggregated and analyzed at community level (a community in a social network is considered to be a group of nodes densely connected internally and sparsely connected externally). The presumption here was that two persons with same personality trait but under distinct sociological influences at community level may react differently and similarly people with different personalities but in different sociological circumstances may react exactly in the same way. However, [Ruan et al.2016] had also discussed that personality is correlated with optimism [Sharpe et al.2011]. Pessimism is principally associated with neuroticism and negative affect while optimism is primarily associated with extraversion and positive affect [Marshall et al.1992]. We draw motivations from all these previous works and to this end this paper reports a comprehensive empirical study to understand how user level optimism/pessimism got influenced by person level psychology i.e. personality and by societal level values & ethics at community level.

We report interesting results and correlations related to influence of psycho-sociological factors on optimists/pessimists on Twitter. A few societal factors were found to to be more influencing than other societal factors. For example, achievement and conformity classes tend to be more influencing than other classes of values in promoting optimistic views in social media. Also a few social factors (or community level factors) like achieve-

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 255–264

ment and stimulation were found to be more positively correlated than user level factors in optimists. On the other hand, for pessimists power-oriented and traditional settings seem to be more correlated than other factors.

The paper is structured as follows. In Section 2, we discuss the relevant research work done in the area of computational models for optimism/pessimism as well as psycho-sociological factors. The data sources used for building classifiers, fuzzy distributions of personality and values in the corpus and performance of classifiers used to infer the distribution of different psycho-sociological aspects are discussed in Section 2. Section 3 describes the methodology used for extracting the semantic representation of a community in terms of it's values. Section 4 explains the use of semantic interpretation of communities in terms of their values and psycho-sociological patterns of users for the calculation of aggregate degree of optimism/pessimism under the influence of different factors. In Section 5, important correlations related to influence of different factors on optimists/pessimists are discussed in detail.

2 Related Work

There has been a little research in the field of computational models to predict the degree of optimism. [Ruan et al.2016] developed the first computational model for measuring the optimism/pessimism of users based on their social media activity. 714 potential optimists and 614 potential pessimists were identified by searching for specific phrases. After identification of potential optimists and pessimists, they created a ground truth dataset through human annotation on a randomly selected subset of corpus. In order to rank the users according to degree of optimism and pessimism, Twitter users were sorted based on the average scores assigned to their tweets.The top 25% were labelled as optimists whereas the bottom 25% were labelled as pessimists.

Deeper understanding of human personality, beliefs, ethics, and values has been a key research agenda in Psychology and Social Science research for several decades. One of the most accepted and widely used frameworks for understanding values is Schwartz Theory of Basic Human Values [Schwartz1992]. Schwartz's 10-Values model postulated and empirically verified (on data obtained from a self-assessment questionnaire) the existence of ten basic Values based on people's motivation. The Schwartz model was proved to be very successful in psychological research as well as in other fields. The ten basic Values are related to various outcomes and effects of a person's role in a society [Argandoña2003]. The Values have also proved to provide an important and powerful explanation of consumer behaviour and how they influence it [Kahle et al.1986].

In the recent years, there have been few initiatives to automatically identify various Big5 Personality traits of individuals from their language usage and behaviour in social media [Goldberg1993]. A milestone in this area was the 2013 Workshop and Shared Task on Computational Personality Recognition[1], repeated in 2014[2]. Further research work in the area of developing computational model for identifying Personality from language usage on Facebook and Twitter has been done in [Park et al.2015] and [Quercia et al.2011] respectively. However, no computational model for *Schwartz' Values* has been tested or examined before.

3 Obtaining Psycho-Sociological Patterns of Users

The psycho-sociological backgrounds of an individual play a very crucial role in determining their inclination of being an optimist or pessimist. In our current work, we analyze two psycho-sociolological aspects in correlation with optimism/pessimism: personality and values. We build two classifiers (by analyzing social media content: tweets and activities) to estimate someone's inclination towards optimism/pessimism:

Personality: [Goldberg1990] to determine which personality types {*openness, conscientiousness, extroversion, agreeableness, neuroticism*} are more inclined towards optimism/pessimism.

Values & Ethics: [Schwartz1992] to estimate the nature of values possessed by optimism/pessimism, in terms of 10 values {*achievement, benevolence, conformity, hedonism, security, self-direction, stimulation, tradition, universalism*} types proposed by Schwartz.

[1] http://mypersonality.org/wiki/doku.php?id=wcpr13

[2] https://sites.google.com/site/wcprst/home/wcpr14

The psycho-sociological patterns thus obtained, can be used as metrics to gauge someone's inclination towards optimism/pessimism. Such finding could be helpful for various practical purposes like online recommendation system, Twitter could improve its "who to follow" suggestions, online advertisement and etc by biasing the random walks procedure used in link prediction algorithms or adding regularization terms in matrix factorization methods.

3.1 Data Sources

We used four datasets for our analysis : (1) Optimism/Pessimism dataset for replication of computational model proposed by [Ruan et al.2016], (2) Personality dataset for developing the computational model for predicting five personality traits at user-level, (3) Values & Ethics dataset for developing a model for prediction of Values & Ethics at user-level and (4) SNAP Dataset for analyzing distribution of optimists/pessimists in influence of different psycho-sociological factors.

Optimism/Pessimism: For the purpose of developing computational models for Optimism/Pessimism, we used the dataset proposed by [Ruan et al.2016]. The Twitter conversations dataset was obtained by searching for key-phrases such as "*I am optimistic*" for identifying optimistic users and keywords such as "hate", "unfair", and "disgust" for identifying potential pessimists. They identified 718 potential optimists and 640 potential pessimists with maximum of $2,000$ tweets per user. A small subset of 500 potential optimists and 500 potential pessimists were selected. To create a human annotated ground truth dataset, 15 tweets for each user were randomly selected. Further, each tweet was annotated as potimist or pessimist.

Personality: The personality labeled gold corpus (10K Facebook status updates of 250 users and their Facebook network properties), released in WCPR'13[3] workshop, was used to build the personality model. From the Table 1, we can observe that the Facebook Personality corpus used in WCPR'13 is balanced corpus with almost equal distribution of users across all five different personality traits.

Values & Ethics: The Values & Ethics data for 367 users was crowd-sourced along with users' Twitter Ids using Amazon Mechanical Turk as a service, while ensuring that the participants came from various cultures and ethnic backgrounds: the participants were equally distributed across the globe – Americans (USA, Canada, Mexico, Brazil), South Asia (India, Pakistan, Bangladesh), and a few East-Asians (Singaporeans, Malaysian, Japanese, Chinese). The selected Asians were checked to be mostly English speaking.

Table 1: Flat distribution of Big5 Personality types in the Facebook Personality corpus: Openness (O), Conscientiousness (C), Extraversion (E), Agreeableness (A), Neuroticism (N), The last column gives the Average Majority Baselines.

O	C	E	A	N	Avg
70.40	52.00	38.40	53.60	39.60	**50.80**

We obtained data from self-assessment based psychometric tests using male/female versions of PVQ, the Portrait Values Questionnaire [Schwartz et al.2001]. The PVQ asks participants to answer each question on a 1–6 Likert rating scale[4]. A rating of 1 means "*not like me at all*" and 6 means "*very much like me*". An example question is "*He likes to take risks. He is always looking for adventures.*" where the user should answer while putting himself in the shoes of "He" in the question. The standard practice is to ask a fixed number of questions per psychological dimension. Therefore, there are five questions for each of the ten Values classes, resulting in a 50 item PVQ questionnaire. Once all the questions in the PVQ have been answered, for each user and for each Values class, a score is generated by averaging all the scores (i.e., user responses) corresponding to the questions in that class, as described by [Schwartz2012]. Further, the rescaling strategy proposed by [Schwartz2012] was used to eliminate randomness from each response given by a user as follows: for each user, the mean response score was first calculated considering all the responses s/he provided, and then the mean score from each response was subtracted. See [Schwartz2012] for more details on PVQ and the score computation mechanism.

The ranges of scores obtained from the previous rescaling method may vary across different Values classes. For instance, the ranges of the rescaled scores for the Twitter Values corpus are as

[3] http://mypersonality.org/wiki/doku.php?id=wcpr13

[4] http://www.simplypsychology.org/likert-scale.html

Table 2: Flat distribution of Schwartz' value types in the corpora: Achievement (AC), Benevolence (BE), Conformity (CO), Hedonism (H), Power (PO), Security (SE), Self-Direction (SD), Stimulation (ST), Tradition (TR), Universalism (UN). The last column gives the Average Majority Baselines.

AC	BE	CO	HE	PO	SE	SD	ST	TR	UN	Avg
81.00	78.70	73.30	77.10	50.10	76.30	83.40	73.60	52.60	82.00	**72.80**

follows: Achievement $[-4.12, 3.36]$, Benevolence $[-1.56, 3.39]$, Conformity $[-3.35, 3.01]$, Hedonism $[-5.18, 4.35]$, Power $[-6.0, 2.27]$, Security $[-2.60, 2.40]$, Self-Direction $[-1.61, 3.40]$, Stimulation $[-5.0, 2.63]$, Tradition $[-4.49, 3.35]$, and Universalism $[-3.33, 3.30]$.[5] Hence the following normalisation formula was applied to move the ranges of the different Values classes to the $[-1, 1]$ interval:

$$x_{scaled} = \frac{2 * (x - x_{min})}{x_{max} - x_{min}} - 1$$

Further, for obtaining text data to train computational model we collected tweets from users' Twitter accounts. However, several challenges have to be addressed when working with Twitter. For example, several users had protected Twitter accounts, so that their tweets were not accessible when using the Twitter API. In addition, many users had to be discarded since they had tweeted less than 100 tweets, making them uninteresting for statistical analysis. In addition, some extreme cases when users mentioned someone else's (some celebrity's) Twitter account, had to be discarded. Finally after filtering the data, we obtained a text dataset of 367 users consisting of at least 100 tweets and maximum of 3200 tweets. Categorical flat distributions of Values types are reported in the Table 2.

SNAP Dataset: In order to analyze the behaviour of optimists/ pessimists at societal level, the egocentric twitter network released by SNAP is used. For the purpose of investigating the sociological factors operating at societal level, 1,562 ground-truth communities spanning over 81306 nodes and $1,768,149$ edges were considered for further analysis (other communities having size less than 5 and with number of tweets less than 100 were discarded).

[5]The distribution of a particular value type over a corpus was analysed using the Bienaymé-Chebyshev Inequality [Bienaymé1853, Tchébichef1867], showing that, for example, most of the Achievement instances (89%) were in the range $[-2.96, 2.84]$.

3.2 Corpus Statistics

Categorical flat distributions of Personality and Values types are reported in Table 1 and Table 2, respectively. It is noteworthy that the Facebook Personality corpus used in the Workshop on Computational Personality Recognition shared task [Celli et al.2013] is quite balanced because it was judiciously chosen from a larger 10K user data corpus collected in the myPersonality project.[6] The shared task organisers chose the right (desired) distribution for the released corpus, but in a real-life setting it is almost impossible to get a balanced data from any user population or social media platform. On the other hand, the Twitter corpus, collected by us is skewed.

Moreover, both the Personality model and Schwartz' Values model support fuzzy membership, which means that anyone having Open personality can have Agreeable nature as well, and similarly that someone with Power orientation also can have Achievement orientation. To understand this notion, the fuzzy membership statistics of Facebook Personality corpus and Twitter Values corpus are reported in Figures 1 and 2 respectively.

On a careful analysis of Figure 1 one can clearly observe how each Personality trait is overlapped with other Personality traits. For example, for Openness (O), we can clearly see that almost equal amount of people are positively oriented towards all the remaining four Personality dimensions. Similarly, Agreeable people are evenly distributed among three traits: Openness, Extroversion and Conscientiousness, but very few of them have in neurotic nature. On the other hand, the class distribution of Neuroticism (N) is highly imbalanced, as most of the people who are neurotic are always eager to experience new things always to satisfy their ever-changing mood. Further, it can be inferred from the visualisation that very few extroverts have neurotic nature, but many of them are positively oriented towards Conscientiousness (C) trait. It indeed makes sense as extrovert people are outgoing and would like to mingle in different circles of the society, and thus they are accommodative and less sentimental, i.e., less neurotic but they are rather methodical i.e., conscientious.

[6]http://mypersonality.org

Personality Traits	O	C	E	A	N
Openness(O)		52.27	40.90	60.22	38.63
Conscientiousness(C)	70.76		46.92	57.69	28.46
Extraversion(E)	75.00	63.54		59.37	22.91
Agreeableness(A)	79.10	55.97	42.53		25.37
Neuroticism(N)	68.68	37.38	22.25	34.56	

Figure 1: Fuzzy distributions of Big5 Personality traits in the myPersonality corpus. Similar to Schwartz's fuzzy distribution this table attempts to present the interconnection between different Personality traits.

Schwartz Values	AC	BE	CO	HE	PO	SE	SD	ST	TR	UN
Achievement(AC)		28.31	19.49	29.41	41.54	15.81	11.77	19.85	41.91	17.28
Benevolence(BE)	24.12		19.84	31.12	52.92	18.68	10.51	22.18	42.8	7
Conformity(CO)	18.59	23.42		35.32	47.58	12.64	17.47	24.91	35.32	15.99
Hedonism(HE)	17.63	24.04	25.32		43.35	21.03	9.01	14.6	45.92	12.88
Power(PO)	12.64	33.52	22.53	27.47		17.58	13.74	17.03	41.21	20.33
Security(SE)	17.63	24.82	15.47	33.81	46.04		13.31	21.94	38.49	14.39
Self-Direction(SD)	21.05	24.34	26.97	30.26	48.35	20.72		20.72	47.04	12.5
Stimulation(ST)	18.66	25.37	24.63	25.75	43.66	19.03	10.08		42.91	16.04
Tradition(TR)	18.13	23.83	9.84	34.72	44.56	11.4	16.58	20.73		17.1
Universalism(UN)	24.75	20.07	24.41	32.11	51.51	20.4	11.04	24.75	46.49	

Figure 2: Fuzzy distributions of Schwartz' values in the Twitter corpus. Schwartz' theory explains how the values are interconnected and influence each other, since the pursuit of any of the values results in either an accordance with one another (e.g., Conformity and Security) or a conflict with at least one other value (e.g., Benevolence and Power).

It is also clear that the fuzziness is much higher among the Values classes than among the Personality traits. One possible reason is that Personality has fewer number of classes than Schwartz Values. Such overlapping nature of psychological classes makes the computational classification problem much more challenging than the classical sentiment analysis problem.

3.3 Psycholinguistic and Network Features

We explored exhaustive set of features including – (**f1**) Word N-grams; (**f2**) Lingustic Features (LIWC[7]; Harvard General Inquirer, MRC psycholinguistic feature; Sensicon[8]); (**f3**) Speech-Act classes; (**f4**) Sentiment Amplifiers (Exclamation Marks, Quotes, Ellipses, Interjections, Emoticons, Word/Sentence Length); (**f5**) Sentiment/ Emotion lexica (NRC emotion Lexicon [Mohammad et al.2013], Sentiwordnet [Esuli and Sebastiani2007]); (**f6**) Topics words obtained from topic model. A brief overview about personality, values, and optimism/pessimism models and features used are illustrated in 3.

Table 3: Features used in psycho-sociological models

Models	f1	f2	f3	f4	f5	f6	F-Score
Personality	+	+	+	-	-	-	79.35%
Values	-	+	+	-	-	+	80.10%
Optimism/Pessimism	+	+	-	-	+	-	77.89%

3.4 Building Classifiers

We collected data from several sources to build three classification models. Here for each model we report the best classifier. The feature engineering required for improving performance of Personality and Values models is discussed in detail in [Tushar Maheshwari2016]. All the results reported in Table 3 are based on 10-fold cross validation on respective datasets.

Personality: The personality labeled gold corpus (10K Facebook status updates of 250 users and their Facebook network properties), released in WCPR'13[9] workshop [Celli et al.2013], is used

[7] http://www.liwc.net/

[8] https://hlt-nlp.fbk.eu/technologies/sensicon

[9] http://mypersonality.org/wiki/doku.php?id=wcpr13

to build the personality model. Our SVM-based model outperforms the state-of-the-art [Verhoeven et al.2013] by 10%, achieving average F-Score of 79.35%. Features used in this model are reported in Table 3.

Values & Ethics: For the values model we crowd-sourced a Twitter corpus using the Amazon Mechanical Turk[10]. Self-assessments were obtained using the Portrait Values Questionnaire (PVQ) [Schwartz et al.2001]. At the end of the data collection process, data from 367 unique users had been gathered, having 1,608 average tweets per user (see supp. for details). The SVM-based values classifiers achieves an average F-Score of 80% using features reported in Table 3.

Optimism/Pessimism: A Multinomial Naive Bayes classifier is trained on the dataset introduced by [Ruan et al.2016], the state-of-the-art for this work. We obtain an F-Score of 77.89% using featues reported in Table 3), which is comparable to the state-of-the-art i.e. 81% [Ruan et al.2016].

4 Semantic Interpretation of Communities

In order to analyze the behaviour of optimists/pessimists at societal level, the egocentric twitter network released by SNAP is used. The Twitter network, released by SNAP [Leskovec and Krevl2015] (nodes: 81,306, edges: 1,768,149) has been used to study community structure. We considered 1,562 ground-truth communities (after discarding communities having size less than 5 and with tweets less than 100).

In order to analyse whether people within the same community tend to be homogeneous with respect to their background values/ethics, we measure Shannon's Entropy (measure of the uncertainty) [Lin1991] for each dimension separately.

Higher entropy scores suggest lower similarity. To calculate the entropy score vector $X_{(i)}$ for a community $C_{(i)}$ consisting of n users as $u_{(1)}, u_{(2)}, u_{(3)}...u_{(n)}$, a matrix $A_{(i)}$ is created where $A_{(i,j)}$ row vector represents the estimated scores of each of the ten values for a user $u_{(j)}$ and $A_{(i,:,k)}$ column vector represents the estimated scores of k^{th} class for all n users. The $A_{(i,:,k)}$ column vector was transformed to a probability distribution vector $N_{(i,:,k)}$ using softmax-normalization:

$$N_{(i,j,k)} = \frac{exp(A_{(i,j,k)})}{||exp(A_{(i,:,k)}||_1}$$

The entropy score $X_{(i,k)}$ for $N_{(i,:,k)}$ can be calculated using the following formulation:

$$X_{(i,k)} = -\sum_{j=1}^{n} N_{(i,j,k)} * \log N_{(i,j,k)}$$

Table 4: Illustrates entropy calculation for values model. Here $T_{(i)}$ represents the binary estimate of fuzzy distribution of values and $S_{(i)}$ represents the zero-mean unit-variance scaled values of $X_{(i)}$ for a community $C_{(i)}$. Similarly, binary estimates for five personality traits $P_{(i)}$ of user $u_{(i)}$ are calculated.

	AC	BE	CO	HE	PO	SE	SD	ST	TR	UN
$u_{(1)}$	0.91	0.47	0.02	0.07	0.32	0.24	0.65	0.78	0.94	0.10
$u_{(2)}$	0.97	0.40	0.49	0.50	0.56	0.83	0.62	0.73	0.04	0.08
$u_{(3)}$	0.99	0.75	0.50	0.72	0.38	0.60	0.75	0.02	0.57	0.62
$u_{(4)}$	0.77	0.44	0.40	0.16	0.19	0.55	0.73	0.08	0.53	0.25
$u_{(5)}$	0.29	0.02	0.26	0.56	0.41	0.23	0.95	0.02	0.79	0.86
$X_{(i)}$	1.54	1.40	1.40	1.39	1.55	1.50	1.59	0.99	1.42	1.28
$S_{(i)}$	0.87	-0.12	-0.12	-0.19	0.95	0.57	1.26	-2.35	0.00	-0.87
$T_{(i)}$	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0

After normalization, $N_{(i,:,k)}$ vector represents the probability distribution of k^{th} Value class across n users where entropy score $X_{(i,k)}$ represents the randomness in community along k^{th} Value class. The lower the randomness, higher the k^{th} class is dominant in the $C_{(i)}$ community. Now, in order to obtain binary estimates $T_{(i)}$ for each of the ten values and classes in $C_{(i)}$ community, the entropy score vector $X_{(i)}$ is scaled using zero-mean unit-variance method and for numerical values greater than 0, 1 was assigned and for numerical values less than 0, 0 was assigned as class label for $C_{(i)}$ community. Instead of labelling a community $C_{(i)}$ with a class having minimum entropy, the scaling approach is used for the purpose of preserving the fuzzy distribution of values at community level. The obtained $T_{(i)}$ vector represents the fuzzy distribution of values and is thus a representation to capture the semantic information about the community.

[10] https://www.mturk.com/

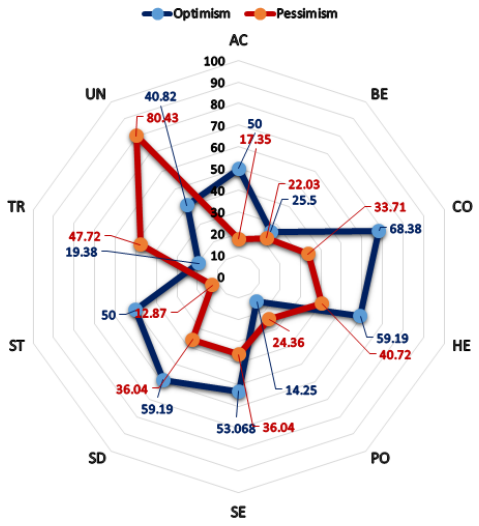


Figure 3: Openness

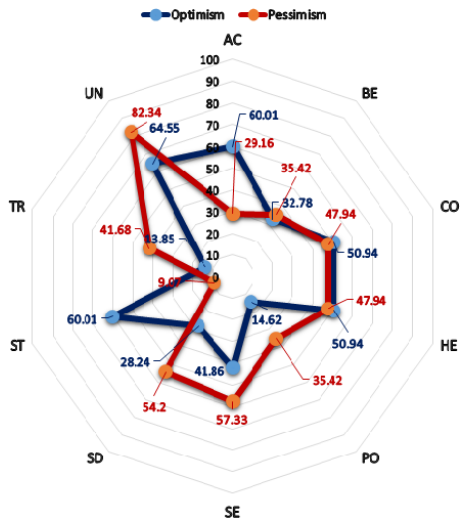


Figure 4: Conscientiousness

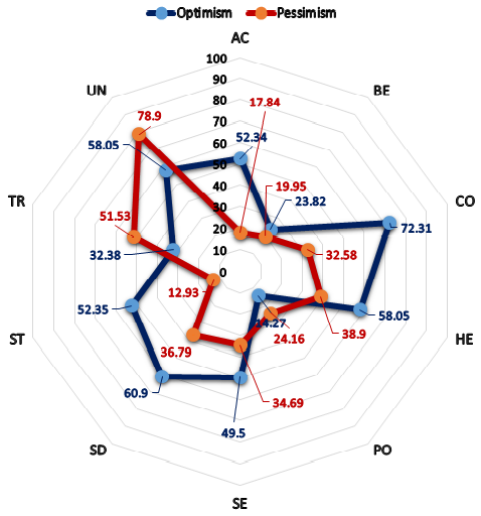


Figure 5: Extraversion

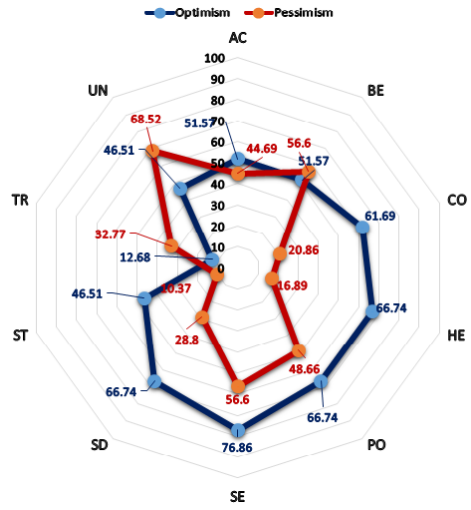


Figure 6: Agreeableness

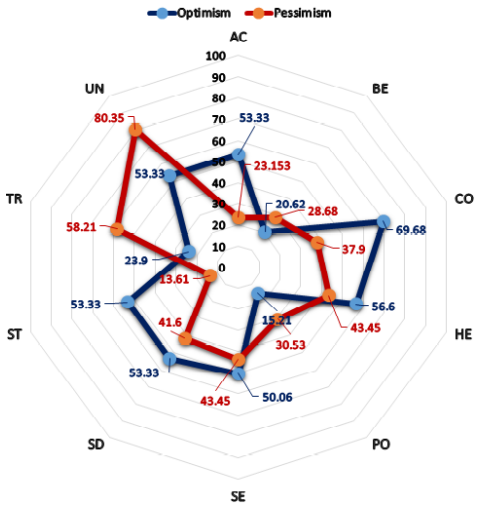


Figure 7: Neuroticism

Figure 8: Psycho-Sociological Influences on Optimism/Pessimism over Twitter. For i^{th} personality trait, the radar chart visualizes $I_{(i,:,0)}$ and $I_{(i,:,1)}$ represents degree of optimism and pessimism respectively over 10 classes of Values & Ethics.

5 Understanding Psycho-Sociological Influences of Optimism/Pessimism

The analysis for user-level estimation starts with each user $u_{(i)}$ and it's the binary estimates of all five traits of the Personality ($P_{(i)}$), probability estimates of ten classes of Values ($V_{(i)}$) and probability estimates of degree of optimism/pessimism ($O_{(i)}$), obtained using the pre-trained Big5 Personality model, Schwartz Values model and optimism/pessimism models respectively. The user-level estimation for the degree of optimism/pessimism was done by averaging the optimism/pessimism scores for each user. Further the estimated scores for five personality traits, ten values traits and degree of optimism/pessimism was scaled using zero-mean unit-variance method for each of the 16 classes independently.

To this extent, we have obtained $P_{(j)}$, $V_{(j)}$ & $O_{(j)}$ representing the personality traits, values and degree of optimism respectively for each user $u_{(j)}$ and binary estimates $T_{(i)}$ representing the fuzzy distribution of values for each community $C_{(i)}$.

Aggregate Analysis over Personality Traits: For the purpose of understanding the distribution of $O_{(j)}$ under the influence of $P_{(j)}$ and $T_{(i)}$ for a user $u_{(j)}$ in community $C_{(i)}$, we divide our analysis in five parts according to five different traits of personality. The division of analysis according to five personality traits will help us better understand the influence of each of these personality traits on optimists/pessimists. Therefore, for all users $u_{(j)}$ having a particular personality trait, we go into the communities the user $u_{(j)}$ belongs to and lookup the values of that community using $T_{(i)}$ as well as obtain the degree of optimism/pessimism for that user from $O_{(j)}$. In this way, we *aggregate the degree of optimism for each pair of personality trait and values class* and obtain the distribution of degree of optimism/pessimism under the influence of socio-psychological factors. The aggregation of degree of optimism/pessimism can be formulated as follows :

$$I_{(i,j,0)} = \sum_{u_{(k)}|(P_{(k,i)}=1)\cap(T_{(k,j)}=1)} O_{(k,j,0)}$$

where $I_{(i,j,0)}$ represents the aggregate degree of optimism for i^{th} personality trait and j^{th} value class. Similarly, $I_{(i,j,1)}$ representing the degree of pessimism is calculated by aggregating over $O_{(k,j,1)}$.

6 Obtained psycho-sociological patterns

On careful investigation of five radar plots (Figure 8) for each of the personality traits we discover interesting patterns on how combination of personality trait and values influence the behaviour of optimist/pessimist users on Twitter.

Influence of Psycho-Sociological Factors on Optimists: In Figure 3, we can infer that people with open personality are generally optimistic irrespective of the dominating values of their community as individuals with high openness tend to seek euphoric experiences resulting in positive expectations. From Figure 4, it is evident that individuals high in conscientiousness are positively correlated with very optimistic people in both achievement as well as stimulation oriented communities. Since, people high in conscientiousness tend to have obsession, they are expected to be optimistic when they are surrounded by people aspiring to achieve and face challenges. From Figure 5, it can be observed that extroversion has positive correlation with very optimistic people in communities highly oriented towards achievement, conformity, hedonism, security, self-direction and stimulation owing to their high energy and positive emotions. Further from Figure 6, we can observe that agreeable people have very distinct radar profile portraying them as very optimistic people irrespective of their social settings because of their compassionate and cooperative nature.

Overall, from the radar charts, we propose the order of dominance of societal factors (values & ethics) over individual factors (personality) by observing that in certain social factors dominate over all user-level factors : AC $>$ CO $>$ ST. For example, achievement and stimulation oriented settings encourage more optimistic views on social media irrespective of different personality traits.

Influence of Psycho-Sociological Factors on Pessimists: Open people in traditional or power oriented settings seem to be more pessimistic owing to many restrictions posed in a traditional community and quest for prestige in a power oriented community. Similarly, for all other personality traits, the traditional settings influence more

than an individual's personality traits in promoting pessimistic views. Except for agreeable people, we can observe that power-oriented community also increases the sense of pessimism among people. In addition, for conscientious users belonging to security or self-direction oriented communities are very pessimistic. The pessimistic nature of the conscientious users can be explained by the fact that their sense of responsibility coupled with pressure for maintaining stability, security and making independent choices in life may lead to lowering their positive expectations.

On the other hand, a few outliers in the analysis can be seen due to accumulation of inaccuracies in the trained models of Values & Ethics as a result of biased training data which is clear from flat distribution of values in Table 2. For example, on an average, optimism in users belonging to communities high in universalism as well as pessimism in users belonging to communities high in conformity is relatively high irrespective of their personality traits.

7 Conclusion and future direction

This paper presents an empirical study to understand *how psycho-sociological factors influence on optimism/pessimism at individual level*.

However, we have only analyzed intra-community psycho-sociological patterns in this study, but we strongly believe that neighbouring communities also have influential roles to play on person level optimism/pessimism. In addition, we need to study the influence of communities which are not power-oriented or not achievement oriented along with different personality traits. We are working on analyzing and inferring other influencing factors using computational models.

References

[Argandoña2003] Antonio Argandoña. 2003. Fostering values in organizations. *Journal of Business Ethics*, 45(1–2):15–28.

[Bienaymé1853] Irénée-Jules Bienaymé. 1853. *Considérations à l'appui de la découverte de Laplace sur la loi de probabilité dans la méthode des moindres carrés*. Imprimerie de Mallet-Bachelier.

[Celli et al.2013] Fabio Celli, Fabio Pianesi, David Stillwell, and Michal Kosinski. 2013. The workshop on computational personality recognition 2013. In *Proceedings of the AAAI*, pages 2–5. AAAI.

[Esuli and Sebastiani2007] Andrea Esuli and Fabrizio Sebastiani. 2007. Sentiwordnet: A high-coverage lexical resource for opinion mining. *Evaluation*, pages 1–26.

[Goldberg1990] Lewis R Goldberg. 1990. An alternative" description of personality": the big-five factor structure. *Journal of personality and social psychology*, 59(6):1216.

[Goldberg1993] Lewis R Goldberg. 1993. The structure of phenotypic personality traits. *American psychologist*, 48(1):26.

[Kahle et al.1986] Lynn R Kahle, Sharon E Beatty, and Pamela Homer. 1986. Alternative measurement approaches to consumer values: The list of values (lov) and values and life style (vals). *Journal of consumer research*, pages 405–409.

[Leskovec and Krevl2015] Jure Leskovec and Andrej Krevl. 2015. {SNAP Datasets}:{Stanford} large network dataset collection.

[Lin1991] Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151.

[Marshall et al.1992] Grant N Marshall, Camille B Wortman, Jeffrey W Kusulas, Linda K Hervig, and Ross R Vickers Jr. 1992. Distinguishing optimism from pessimism: Relations to fundamental dimensions of mood and personality. *Journal of personality and social psychology*, 62(6):1067.

[Mohammad et al.2013] Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*.

[Park et al.2015] Gregory Park, H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Michal Kosinski, David J Stillwell, Lyle H Ungar, and Martin EP Seligman. 2015.

Automatic personality assessment through social media language. *Journal of personality and social psychology*, 108(6):934.

[Quercia et al.2011] Daniele Quercia, Michal Kosinski, David Stillwell, and Jon Crowcroft. 2011. Our twitter profiles, our selves: Predicting personality with twitter. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 180–185. IEEE.

[Ruan et al.2016] Xianzhi Ruan, Steven R Wilson, and Rada Mihalcea. 2016. Finding optimists and pessimists on twitter. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 320.

[Schwartz et al.2001] Shalom H Schwartz, Gila Melech, Arielle Lehmann, Steven Burgess, Mari Harris, and Vicki Owens. 2001. Extending the cross-cultural validity of the theory of basic human values with a different method of measurement. *Journal of cross-cultural psychology*, 32(5):519–542.

[Schwartz1992] Shalom H. Schwartz. 1992. Universals in the content and structure of values: theoretical advances and empirical tests in 20 countries. In *Advances in Experimental Social Psychology*, pages 1–65. In M. Zanna (Ed.), Advances in experimental social psychology (Vol. 25), New York: Academic Press.

[Schwartz2012] Shalom H Schwartz. 2012. An overview of the schwartz theory of basic values. *Online Readings in Psychology and Culture*, 2(1):11.

[Sharpe et al.2011] J Patrick Sharpe, Nicholas R Martin, and Kelly A Roth. 2011. Optimism and the big five factors of personality: Beyond neuroticism and extraversion. *Personality and Individual Differences*, 51(8):946–951.

[Tchébichef1867] Pafnuty Lvovich Tchébichef. 1867. Des valeurs moyennes (translated into French by N.V. Khanykov). *Journal de Mathématiques Pures et Appliquées*, 12(2):177–184.

[Tushar Maheshwari2016] Upendra Kumar Amitava Das Tushar Maheshwari, Aishwarya Reganti. 2016. Semantic interpretation of community in social networks. *AAAI*.

[Verhoeven et al.2013] Ben Verhoeven, Walter Daelemans, and Tom De Smedt. 2013. Ensemble methods for personality recognition. *Proceedings of WCPR13, in conjunction with ICWSM-13*.

End to End Dialog System for Telugu

Prathyusha Danda[1] **Prathyusha Jwalapuram**[2] **Manish Shrivastava**[3]
Language Technologies Research Center
Kohli Center on Intelligent Systems
International Institute of Information Technology
Hyderabad, India
[1]danda.prathyusha@research.iiit.ac.in
[2]prathyusha.jwalapuram@research.iiit.ac.in
[3]m.shrivastava@iiit.ac.in

Abstract

This paper describes an end to end dialog system created using sequence to sequence learning and memory networks for Telugu, a low-resource language. We automatically generate dialog data for Telugu in the tourist domain, using a knowledge base that provides tourist place, type, tour time, etc. Using this data, we train a sequence to sequence model to learn system responses in the dialog. In order to add the query prediction for information retrieval (through API calls), we train a memory network. We also handle cases requiring updation of API calls and querying for additional information. Using the combination of sequence to sequence learning and memory network, we successfully create an end to end dialog system for Telugu.

1 Introduction

There have been few attempts to create dialog systems for Telugu, which are mostly rule-based systems using ad-hoc user interactions to test the system rather than over a set of prepared test dialogs (Sravanthi et al., 2015; Reddy and Bandyopadhyay, 2006). This is primarily due to a lack of dialog data as Telugu is a low-resource language. Wen et al. (2016) proclaim that the greatest bottleneck for statistical approaches to dialog system development is the collection of appropriate data which is especially true for task oriented dialog systems; that for task-oriented dialog systems, in-domain data is essential.

Dialog models using neural networks are able to leverage the large amounts of data to learn meaningful representations for natural language and generation strategies, and require only a minimal amount of domain knowledge and handcrafting (Serban et al., 2016). The neural networks are used to represent both dialog histories and to produce output either through a generative model that generates responses word-by-word conditioned on a dialog context (which is the model this paper uses) or through a discriminative model that is trained to select an appropriate response from a set of candidate responses (Serban et al., 2016). We use both the models for generating system responses in our dialog system.

Sequence to Sequence learning (Sutskever et al., 2014) has been used to build end-to-end trainable non-task-oriented conversational dialog systems (Vinyals and Le, 2015; Shang et al., 2015; Serban et al., 2015). This approach models dialog as a source to target sequence transduction problem, applying an encoder network (Cho et al., 2014) to encode a user query into a distributed vector representation of its semantics, which conditions a decoder network to generate each system response. This has been extended to a task-oriented system that interacts with a knowledge base by Wen et al. (2016).

End-to-end dialog systems are trained on past dialogs directly, with no assumptions made on the basis of the domain or on the structure of the dialog, which makes scaling up automatically to new domains easy (Bordes and Weston, 2017). As an end-to-end neural model, Memory Networks (Weston et al., 2015a), with an attention based architecture, showed promising results for non goal-oriented dialog (Dodge et al., 2016), and have also been applied to question answering (Weston et al., 2015b; Bordes et al., 2015) and language modelling (Sukhbaatar et al., 2015). However, goal-oriented dialog requires the system to ask questions to clearly define a user request, query knowledge bases, etc., as extended by Bordes and Weston (2017).

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 265–272

We first create a corpus of Telugu dialog data in the Tourist domain, which we then use to train our sequence to sequence and memory network models. We report our results for system response generation through the sequence to sequence model, and our results for API call generation, for retrieving information from knowledge base, through the memory network model. Through this combination of sequence to sequence learning and memory network, we successfully create an end-to-end dialog system for the tourist domain in Telugu.

After discussing Related Work in Section 2, we outline the tasks our system must perform in Section 3, then we discuss the motivation behind our system pipeline in Section 4, Section 5 describes dialog data creation strategy, followed by sequence-to-sequence model for producing system responses in Section 6, Section 7 deals with the memory network layer for generating API calls, finally followed by the conclusion and future work in Sections 8 and 9 respectively.

2 Related Work

Ritter et al. (2011) first proposed using generative probabilistic models to model conversations from micro-blogging websites, treating the response generation problem as a statistical machine translation problem, where the post is to be translated into a response. They find that generating responses is a harder problem than language translation due to the wide range of possible responses and a lack of alignment between the source and the response.

Shang et al. (2015) extend the work by using recurrent neural networks, which they show outperform retrieval-based and SMT based methods for generating responses to a post in Chinese, and are able to generate multiple responses with variety. Sordoni et al. (2015) go further by designing the response generation to be conditioned on past dialog utterances that provide contextual information, and also outperform MT and IR based models.

The end-to-end trainable, non-task-oriented conversational dialog systems built by Vinyals and Le (2015; Shang et al. (2015; Serban et al. (2015) using sequence to sequence learning (Sutskever et al., 2014) are promising chatbot systems but do not support domain specific tasks and do not interact with knowledge bases such as databases (Sukhbaatar et al., 2015; Yin et al., 2015), and therefore cannot provide useful information through their responses.

Wen et al. (2016) augment the sequence to sequence architecture with dialog history modelled by a set of belief trackers, and a distributed representation of user intent with delexicalisation and weight tying strategies. Their system provides relevant and appropriate responses at each turn and also interacts with a database through a slot-value pair representation of attributes. They achieve a high task success rate and show that the learned model can interact efficiently and naturally with human subjects to complete an application specific task.

Dodge et al. (2016) use Memory Networks (Weston et al., 2015a; Sukhbaatar et al., 2015) to train non goal oriented dialog, which showed promising results. Bordes and Weston (2017) train memory networks to perform tasks non-trivial tasks such as issuing API calls to knowledge bases and manipulating entities unseen in training; the bot is also able to ask questions to fill missing information. They show that memory networks can outperform a dedicated slot-filling rule-based baseline, and even classical IR and supervised embeddings; they solve the task of issuing API calls.

3 Tasks

As part of our dialog system, there are four main tasks we want to accomplish:

1. Generate appropriate system responses to user utterances.
2. Generate API calls for information retrieval
3. Generate API calls for information retrieval in case of updation.
4. Generate API calls for providing additional information

API calls represent specific operations that applications can invoke at runtime to perform tasks, one of which is querying data from the knowledge base[1]. In our implementation, the API calls simply consist of the keywords that we use to query the knowledge base to retrieve the highest rated tourist location or to give additional information such as address or phone number or opening time of a tourist location. Additional information (Task 4) consists of phone number, address and opening times.

[1] https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/calls.htm

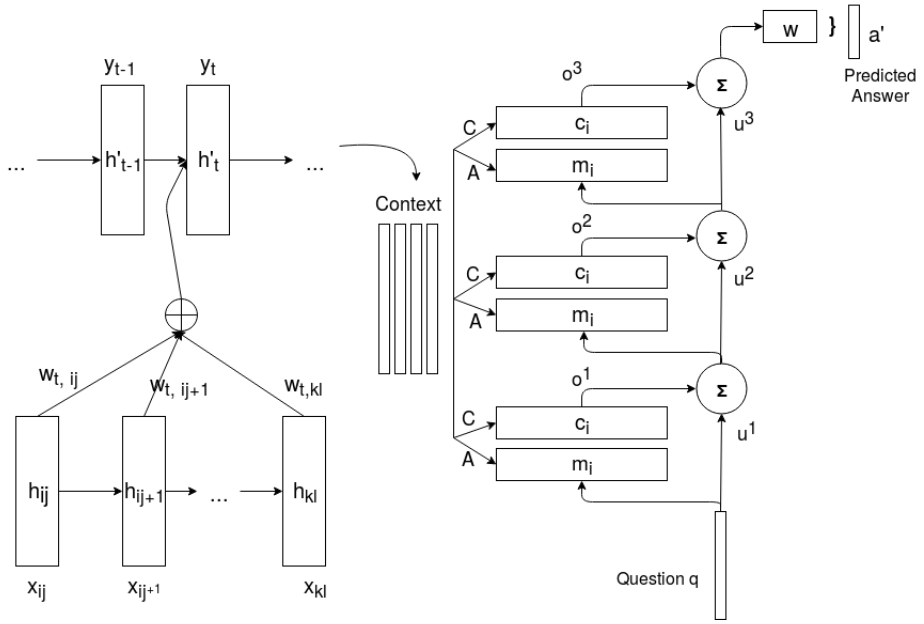


Figure 1: Network Architecture

4 Motivation behind System Pipeline

A sequence to sequence model generates a sentence by generating a sequence of words whereas in memory networks a system response is generated by picking one from all the possible dialog candidates, mentioned in Bordes and Weston (2017).

Hence, predicting system responses using a sequence to sequence model is preferable over memory networks. (See Figure 2)

5 Data Creation

In order to automatically create the substantial amount of data required to train an end-to-end dialog system, we use an approach similar to the one used by Bordes and Weston (2017). In Bordes and Weston (2017), data is simulated based on an underlying restaurant domain knowledge base that has attributes such as type of cuisine, location, etc. and can be queried using API calls.

Our knowledge base is similarly built on the tourist domain, and has the attributes area, type, tour duration, opening time, rating, phone number and address[2], of which area, type and tour duration are the 3 required keywords while opening time, phone number and address are query-able fields. Areas consist of place-names (such as Ban-

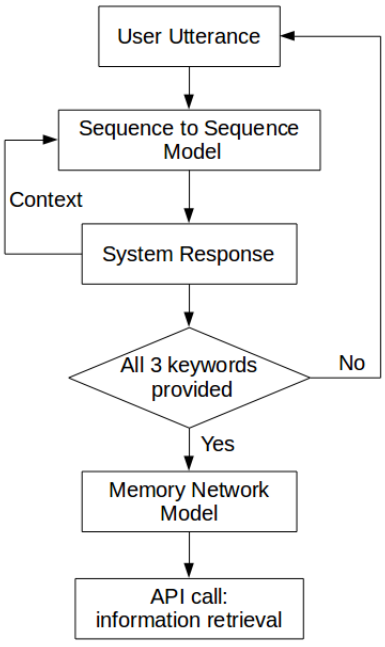


Figure 2: End to End System Pipeline

[2]https://en.wikipedia.org/wiki/List_of_tourist_attractions_in_Hyderabad

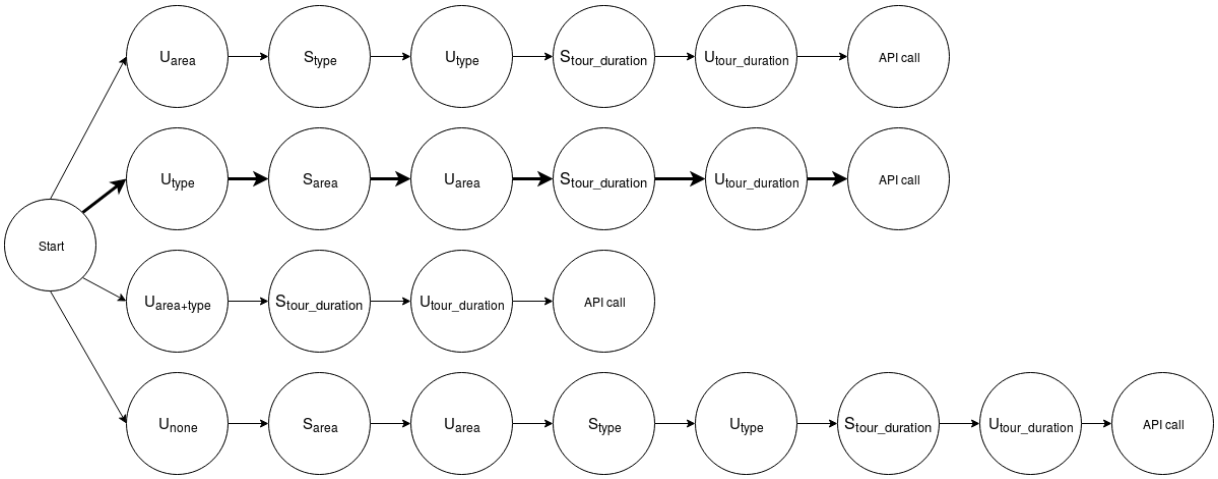


Figure 3: System Dialog Flow

jara Hills, Chanda Nagar Village, etc.) and types consist of types of tourist locations such as historical, religious, zoo and amusement park. The tour duration is the time in which the user wants to see a place, such as 15 minutes, 30 minutes, etc. There are a total of 30 areas, 4 types and 4 tour durations.

Based on this knowledge base, we generate queries by choosing any of the three required fields for a query, which are area, type and tour duration. Using natural language patterns in Telugu, we create user and system utterances. We append the keywords required for a query to the knowledge base at the end of each dialog in the form of an API call (e.g. api_call banjara hills zoo 15, api_call banjara hills zoo 15 address). However, since the knowledge base is in English, the API calls are also in English.

There are 68 possible patterns for the user to express 12 different intents and 9 possible patterns for the bot and a total of 1920 possible API calls. Although all the words in the language patterns are in Telugu, certain words which are commonly used in English are also transcribed and included, helping us handle some code-mixed cases.

Different permutations of the patterns combined with different entities in the knowledge base produce thousands of dialogs, which also include the API call that is required for information retrieval. See Figure 4 for an example of the dialog created for Tasks 1 and 2.

The user could begin by providing no information (U_{none}), which should prompt the system to ask for the area (S_{area}); the user could provide the area in the first utterance (U_{area}), which should prompt the system to ask for the type (S_{type}); the user may provide both the area and the type in the first utterance itself ($U_{place+type}$), which should prompt the system to ask for the tour duration ($S_{tour\_duration}$), and so on, and finally the system must be able to generate the correct corresponding API call to retrieve an appropriate tourist location or to give address or phone number of a particular tourist place (Task 1, 2 and 4).

The user can always request a change in the required fields, which should lead to an update in the API call. This forms a separate dataset for Task 3.

The possible paths are given in an example diagram Figure 3; the path taken in the example dialog in Figure 4 is highlighted.

The order of the fields can be jumbled through the utterances, but the system queries unfilled slots deterministically, i.e., the user may begin by specifying the type of location they want to visit and then be prompted to provide the area, whereas system will prompt for area, type and tour time in that order. Through this, we have 4 variations of the dialog beginnings: user specifying area first, user specifying type first, user specifying both area and type, and user specifying neither.

We partition the data differently (therefore producing differing amounts) for training the sequence to sequence model and the memory networks; this is described in their respective sections. The length of the dialogs varies from 5 to 12 system-user utterance pairs depending on the combinations. The maximum possible length of an utterance is 14.

We have released the data for public use so that those who wish to use the Telugu data for research

User Utterance	System Response
నేను హిస్టారికల్ ప్లేస్ చూడాలని అనుకుంటున్నాను *nEnu histArikal plEs cUdAlani anukuMtunnAnu* I want to visit a historical place.	సరే. ఎక్కడ ఉండాలి *sarE. ekkaduMdAli* Okay. Where should it be?
నేను కాంచన బాగ్ లో ఉన్న ప్రాంతాలు చూడాలని అనుకుంటున్నాను *nEnu kAMcana bAg lO unnA prAMwAlu cUdAlani anukuMtunnanu* I want to visit the places in Kanchanbagh.	సరే. ఎంత సమయం లో చూడాలి అనుకుంటున్నారు *sarE. eMwa samayaM lO cUdAli anukuMtunnAru* Okay. How long do you want to spend visiting?
15 నిమిషాలలో చూడాలి అనుకుంటున్నాను *15 nimiRAlalO cUdAli anukuMTunnAnu* I want to spend 15 minutes.	సరే ఏ ప్రదేశాలు ఉన్నాయో చూసి చెప్తాను ఆగండి *sarE. E praxESAlu unnAyO cUsi cepwAnu AgaMdi* Okay. I will check which places are available and let you know. api_call kanchanbagh historical 15

Figure 4: Example Dialog Created

can do so.[3]

6 System Response Generation Using Sequence to Sequence Learning

6.1 Training and Testing Data

In order to train a sequence to sequence learning model that can incorporate dialog history, the dialog data is partitioned in such a way as to provide instances with differing amounts of context. This means that to predict system response S_t, the input with context is $U_1, S_1, ..., S_{t-1}, U_t$. See Figure 5 for an example.

In each case, the queries are ordered deterministically; system must recognize the missing information and ask the user to provide it accordingly.

Through such partitioning, we obtain 180,000 instances of dialog with context. We sample 20,000 instances for training and separate 3,000 instances for testing for Task 1 and Task 2, and the same number of instances for Task 3 from its own dataset.

6.2 Architecture

We use an Encoder-Decoder (Sutskever et al., 2014) architecture for sequence to sequence learning, that uses one GRU (Cho et al., 2014) layer to encode the input sentence one timestep at a time to obtain a large fixed-dimensional vector representation, and then uses another GRU with attention to decode (Bahdanau et al., 2014) the output sequence for that vector.

In Figure 1, x_{ij} corresponds to the one-hot vector of j^{th} word in i^{th} sequence. The h_{ij} is a vector embedding x_{ij}. The $w_{t,ij}$ is the attention weight of t^{th} word in the output sequence corresponding to h_{ij}. y_t is the t^{th} target word in the output sequence, with h'_{t-1} being the RNN hidden state.

The embedding layer over the encoder takes a $|V_{in}|$ sized vector and outputs a vector of hidden size, where $|V_{in}|$ is the size of the input vocabulary and the hidden size is 256.

The loss function we use is negative log likelihood. The model reaches peak accuracy before 10 epochs.

6.3 Experiments

We conduct five experiments with the sequence to sequence model:

1. System response and API call prediction without context (Task 1 and 2)
2. System response and API call prediction with context (Task 1 and 2)
3. System response prediction with context for, where only API call occurrence is predicted, not the API call itself (Task 1)
4. System response and API call prediction with context for updation (Task 1 and 3)
5. System response prediction with context for updation, where only API call occurrence is

[3]https://github.com/dandaprathyusha/End-to-end-dialog-system-for-telugu

Instance	User Utterance	System Response
Utterances in Context = 1	నాకు హయత్నగర్ లో ఉన్న ప్రాంతాలు చూపించు nAku hayawnagar lO unnA prAMwAlu cUpincu Show me the places in Hayathnagar.	<to predict>
Utterances in Context = 3	నాకు హయత్నగర్ లో ఉన్న ప్రాంతాలు చూపించు nAku hayawnagar lO unnA prAMwAlu cUpincu Show me the places in Hayathnagar. నాకు హిస్టారికల్ ప్లేస్ చూడాలని ఉంది nAku histArikal plEs cUdAlani uMdi I feel like visiting a historical place.	సరే ఎలాంటి ప్రాంతాలు చూడాలని అనుకుంటున్నారు sarE elAnti prAMwAlu cUdAlani anukuMTunnAru Okay. What kind of places do you want to see? <to predict>
Utterances in Context = 5	నాకు హయత్నగర్ లో ఉన్న ప్రాంతాలు చూపించు nAku hayawnagar lO unnA prAMwAlu cUpincu Show me the places in Hayathnagar. నాకు హిస్టారికల్ ప్లేస్ చూడాలని ఉంది nAku histArikal plEs cUdAlani uMdi I feel like visiting a historical place. 60 నిమిషాలలో చూడాలి అనుకుంటున్నాను 60 nimiRAlalO cUdAlani anukuMtunnAnu I want to spend 60 minutes.	సరే ఎలాంటి ప్రాంతాలు చూడాలని అనుకుంటున్నారు sarE elAnti prAMwAlu cUdAlani anukuMTunnAru Okay. What kind of places do you want to see? సరే ఎంత సమయం లో చూడాలి అనుకుంటున్నారు sarE eMwa samayaM lo cUdAlani anukuMtunnAru Okay. How long do you want to spend? <to predict>

Figure 5: Examples of Varying Context Length

predicted, not the API call itself (Task 1 and 3)

In the third and fifth experiments, we replace the full API call (api_call kanchanbagh historical 15) with just *api_call*, which the sequence to sequence model learns as a placeholder.

6.4 Results

No.	Experiments	Acc.
1	Without Context + API calls	59.8%
2	Context + API calls	85.54%
3	Context Without API calls	100%
4	Context + API calls	79.67%
5	Context Without API calls	100%

Table 1: Sequence to Sequence Experiment Results

We can see from the results in Table 1 (the first column refers to experiment number) that the addition of context improves the accuracy. On analyzing the system utterances predicted during the second experiment, we saw that sequence to sequence learning is quite poor at predicting the required API calls. This is possibly due to the varying lengths in place names, etc. due to which the model is unable to predict all the components of an API call.

The third and fifth experiments which showed that the 14.46% error in the second experiment and the 20.33% error in the fourth experiment is mainly due to errors in predicting API calls, since their removal results in complete accuracy.

7 Predicting API calls using Memory Network

Since the sequence to sequence model performs poorly in API call prediction, we use memory networks to learn the same.

API calls are specific operations for information retrieval; we can consider predicting them as a simple classification problem. Memory networks are therefore more suitable for this task than a sequence-sequence model. They are unaffected by the varying lengths of place names unlike the sequence to sequence model.

The API call predicted by the Memory network is used to retrieve the required information from the knowledge base; typically a tourist location which fits the criteria in the query, with the highest rating.

7.1 Training and Testing Data

For Task 2 in memory networks, the input consists of a complete dialog history that has all the three fields of area, type and tour duration that are required for the completion of the query, in any

order, up to, but not including, the *api_call* placeholder. The API call placeholder will be replaced by the API call predicted using the memory network.

In order to maintain an equal distribution of different combinations of queries (user specifies area first, user specifies type first, user specifies both area and type, user specifies none), we sample 24,000 instances of dialogs (6,000 of each type), of which we separate 20,000 instances (5,000 of each type) for training and 4,000 (1,000 of each type) instances for testing.

For Task 3, the input starts from the first predicted API call up to, but not including, the final *api_call* placeholder. The final API call placeholder will be replaced by the API call predicted using the memory network. For this task, we sample 15,000 instances for training and 3,000 for testing.

Task 4 is run only on memory network. The input starts from the last predicted API call up to the user's query for additional information. We train on 15,000 instances and test on 3,000.

7.2 Architecture

We use the architecture described by Sukhbaatar et al. (2015), which is primarily a recurrent neural network (RNN) which reads from an external memory before outputting a symbol. (See Figure 1)

A sequence of user (U_i) and system(S_i) utterances $U_1, S_1, U_2..., S_{n-1}$ are taken as memory input and the last user utterance U_n, which is the query q, whose corresponding system response is an API call, is actual label a. The answer that will be predicted a' by our model is the system response: API call.

In our model we are using layer-wise weight-tying where the input and the output embeddings are the same across different layers, i.e. $A^1 = A^2 = ... = A^K = A$ and $C^1 = C^2 = ...C^K = C$. The matrices A, C, of size $d \times V$, and the final weight matrix W, of size $V \times d$, are jointly learnt by minimizing a standard cross-entropy loss, where embedding dimension d is 150.

The memory network reaches peak accuracy within 100 epochs for both tasks.

7.3 Results

In Table 2, Experiment 1 corresponds to Task 2, Experiment 2 corresponds to Task 3 and Experiment 3 corresponds to Task 4. Our accuracy for Task 2 and 3 in predicting API calls is on par with Bordes and Weston (2017). We perform better than Bordes and Weston (2017) in Task 4 since we do not add knowledge base facts, corresponding to the tourist location in the last API call, to the dialog history.

No.	Experiment	Acc.
1	API calls	100%
2	Updated API calls	99.93%
3	API calls for Add. Info.	100%

Table 2: Memory Network Experiment Results

Since the data conforms to certain templates, the memory network performs very well. The accuracy is likely to drop for real world data.

8 Conclusion

We create a fairly large corpus of Telugu dialog data that can be used to train data-intensive models like neural networks, and can be used for further research.

We use the data to train a sequence-to-sequence dialog model that performs very well on predicting system responses, although it fares poorly with predicting API calls for information retrieval. Using memory networks we solve the API call prediction. We retrieve the highest rated tourist locations or other additional information from the knowledge base using the API call.

Our system is the only end-to-end dialog system to use deep learning methods in Telugu and proposes a better and more flexible model than the existing rule-based dialog system by Sravanthi et al. (2015; Reddy and Bandyopadhyay (2006), which can handle very few patterns. Our system is on par with the end-to-end dialog systems for English.

We have used a knowledge base in English and are able to predict API calls in English despite the dialogs being in Telugu. This means that we can use existing knowledge bases in English to build dialog systems in other low resource languages using similar methods of dialog data generation and deep learning.

9 Future Work

The system can be improved by introducing more initiative, for example, providing suggestions, taking negative responses into account, etc. The system should also be able to handle cases where no

results are found for the user query by giving alternatives.

The system must also be trained with more varied data which has a greater number of patterns occurring; ideally on a sizeable corpus of natural dialog data created by native speakers. The system can then be tested subjectively through human evaluators.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *Proceedings of ICLR*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. *Proceedings of ICLR*.

Rami Reddy Nandi Reddy and Sivaji Bandyopadhyay. 2006. Dialogue based question answering system in telugu. In *Proceedings of the Workshop on Multilingual Question Answering*, pages 53–60. Association for Computational Linguistics.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2015. Hierarchical neural network generative models for movie dialogues. *CoRR, abs/1507.04808*.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2016. Generative deep neural networks for dialogue: A short review. *arXiv preprint arXiv:1611.06216*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *Association for Computational Linguistics*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 553–562.

Mullapudi Ch Sravanthi, Kuncham Prathyusha, and Radhika Mamidi. 2015. A dialogue system for telugu, a resource-poor language. In *CICLing (2)*, pages 364–374.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015a. Memory networks. *Proceedings of ICLR*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015b. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables. *arXiv preprint arXiv:1512.00965*.

Investigating how well contextual features are captured by bi-directional recurrent neural network models

Kushal Chawla[1*], **Sunil Kumar Sahu**[2*], **Ashish Anand**[3]
[1]Adobe Research, Big Data Experience Lab, Bangalore, Karnataka, India
[2]National Center for Text Mining, The University of Manchester, United Kingdom
[3]Department of Computer Science and Engineering, IIT Guwahati, Assam, India
kchawla@adobe.com
sunil.sahu@manchester.ac.uk
anand.ashish@iitg.ernet.in

Abstract

Learning algorithms for natural language processing (NLP) tasks traditionally rely on manually defined relevant contextual features. On the other hand, neural network models using an only distributional representation of words have been successfully applied for several NLP tasks. Such models learn features automatically and avoid explicit feature engineering. Across several domains, neural models become a natural choice specifically when limited characteristics of data are known. However, this flexibility comes at the cost of interpretability. In this paper, we define three different methods to investigate ability of bi-directional recurrent neural networks (RNNs) in capturing contextual features. In particular, we analyze RNNs for sequence tagging tasks. We perform a comprehensive analysis on general as well as biomedical domain datasets. Our experiments focus on important contextual words as features, which can easily be extended to analyze various other feature types. We also investigate positional effects of context words and show how the developed methods can be used for error analysis.

1 Introduction

Learning approaches for NLP tasks can be broadly put into two categories based on the way features are obtained or defined. The traditional way is to design features according to a specific problem setting and then use appropriate learning approach. Examples of such methods include classification algorithms like SVM (Hong, 2005) and CRF (Lafferty et al., 2001) among others for several NLP tasks. A significant proportion of overall effort is spent on feature engineering itself. The desire to obtain better performance on a particular problem makes the researchers come up with a domain and task-specific set of features. The primary advantage of using these models is their interpretability. However, dependence on handcrafted features limits their applicability in low resource domain where obtaining a rich set of features is difficult.

On the other hand, neural network models provide a more generalised way of approaching problems in NLP domain. The models can learn relevant features with minimal efforts in explicit feature engineering. This ability allows the use of such models for problems in low resource domain.

The primary drawback of neural network models is that they are too complicated to interpret as the features are not manually defined. Neural networks have been applied significantly to various tasks without many insights on what the underlying structural properties are and how the models learn to classify the inputs correctly. Mostly inspired by computer vision (Simonyan et al., 2013; Nguyen et al., 2015), several mathematical and visual techniques have been developed in this direction (Elman, 1989; Karpathy et al., 2015; Li et al., 2016).

In contrast to the existing works, this study aims to investigate ability of recurrent neural models to capture important context words. Towards this goal, we define multiple measures based on word erasure technique (Li et al., 2016). We do a comprehensive analysis of performance of bi-directional recurrent neural network models for sequence tagging tasks using these measures.

*Part of this work was done while authors were students at IIT Guwahati.

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 273–282

Analysis is focused at understanding how well the relevant contextual words are being captured by different neural models in different settings. The analysis provides a general tool to compare between different models, show that how neural networks follow our intuition by giving importance to more relevant words, study positional effects of context words and provide error analysis for improving the results.

2 Proposed Methods

A sequence tagging task involves assigning a tag (from a predefined set) to each element present in a given sequence. We model Name Entity Recognition (NER) as a sequence tagging task. We follow BIO-tagging scheme, where each named entity type is associated with two labels, $B-entity$ (standing for *Beginning*) and $I-entity$ (standing for *Intermediate*). The BIO scheme uses another label O(standing for *Other*) for all the context or non-entity words.

In this section, we discuss three methods to calculate the importance score of context words. Each method creates a different ranking of context words corresponding to each entity type for a given dataset. The methods range from simple frequency based to considering sentence level or individual word level effects. We assume that we have a pretrained model M on a given dataset.

2.1 Based on word frequency

For a given sentence $S \in$ test set D, consider a window of a particular size around each entity phrase (single or multi word, defined by true tags) w_e in S. We increment the score (corresponding to w_e's entity type e only) for each of the context words present in this window by one. For instance, the CoNLL-2003 shared task data (described in section 3.2) has 4 entity types, namely, *organization* (ORG), *location* (LOC), *person* (PER) and *miscellaneous* ($MISC$). The corresponding labels under BIO-tagging scheme are *B-ORG*, *I-ORG*, *B-LOC*, *I-LOC* and so on. For a 2-word phrase with true tags as (*B-LOC*, *I-LOC*), the score corresponding to LOC for each context word (with true tag as O) in the window is incremented by one. Let the score for a context word w_c corresponding to entity type e in one sentence be $A(w_c, e, S)$.

Hence the relevance score is calculated as follows:

$$I(w_c, e) = \frac{\sum\limits_{\forall S \in D} A(w_c, e, S)}{\sum\limits_{\forall w_c} \sum\limits_{\forall S \in D} A(w_c, e, S)} \quad (1)$$

Using inverse frequency to account for irrelevant, too frequent words, the score can be calculated as follows:

$$I(w_c, e) \quad (2)$$
$$= \left(\frac{\sum\limits_{\forall S \in D} A(w_c, e, S)}{\sum\limits_{\forall w_c} \sum\limits_{\forall S \in D} A(w_c, e, S)} \right) \left(\frac{\sum\limits_{\forall e'} \sum\limits_{\forall w_c} \sum\limits_{\forall S \in D} A(w_c, e', S)}{\sum\limits_{\forall e'} \sum\limits_{\forall S \in D} A(w_c, e', S) + k} \right)$$

where k accounts for 0 counts and sum over e' means summing over all the remaining entity types. In our experiments, we use k=1 and a window size of 11 (5 words on each side). We refer to these methods collectively as M_WF in rest of the paper.

2.2 Using sentence level log likelihood

In the M_WF method, the relevance of each context word is calculated irrespective of its dependence on other words in the sentence. We define another measure using sentence level log likelihood to take into account the dependency between words in a sentence. We refer to this method as M_SLL in rest of the paper.

Let the set of all context words be W and that of all entity types be E. Define $S_{w_c,e}$ as the set of all sentences where both the word $w_c \in W$ and entity type $e \in E$ are present. We say that an entity type e is present in a sentence S, if $\exists$ a word $\in S$ which has it's true tag corresponding to entity type e. Let $F(w_c, e)$ be the size of set $S_{w_c,e}$.

Now, let the true tag sequence for a sentence S be S_{TAGS}. For a context word $w_c \in S$, let $L_1(w_c, S)$ be the negative log likelihood of S_{TAGS} obtained from pretrained model M. Note that since we are working at a sentence level, $L_1(w_c, S)$ will be same for all the context words and entities present in S.

We adapt the erasure method of Li et al. (2016). Here, we replace the representation of word w_c with a random word representation having same number of dimensions and recalculate the negative log likelihood for the true tag sequence S_{TAGS}. Let this value be $L_2(w_c, S)$. Intuitively, if $S \in S_{w_c,e}$ and w_c is relevant for the entity type e, the probability of the true sequence should decrease when the word is removed from the sentence. Correspondingly, it's negative log likelihood value

should increase. Hence, the score $I(w_c, e)$ for a given word corresponding to the entity type can be calculated in the following manner:

$$I(w_c, e) = \frac{1}{F(w_c, e)} \sum_{\forall S \in S_{w_c,e}} \frac{L_2(w_c, S) - L_1(w_c, S)}{L_1(w_c, S)} \quad (3)$$

2.3 Considering left and right word contexts separately

The relevance scoring method M_SLL does not distinguish between words present in the same sentence. The third method, referred to as M_LRC, works at word level and calculates relevance score of each word by distinguishing its presence in the left or right side of the entity word. The measure is defined in a way that it does take into account of dependency between words in the sentence. In a bi-directional setting, the hidden layer representation for any word in a sentence, is a concatenation of two representations - one which combines words to the left, and the other which combines the words to the right.

In the output layer, we combine the weight parameters and the hidden layer representation by a dot product. We divide this dot product in two parts as discussed below. Say the hidden representation is h and weight parameters corresponding to a tag t $\in T$ (set of all possible tags) are represented by p_t. We can write the dot product $p_t^T h$ as a sum of two dot products $p_{t,L}^T h_L$ and $p_{t,R}^T h_R$, representing the contribution from left and right parts separately. In our experiments, we also include the bias term as a weight parameter.

Now, take a sentence S, a context word w_c in S, and an entity word w_e in S with true tag $t \in T$ corresponding to entity type $e \in E$. Define $AvgSum(w_c, w_e, S)$ as follows:

$$AvgSum(w_c, w_e, S) = \frac{\sum_{\forall f \in T - \{t\}} p_{f,K}^T h_K}{\alpha} \quad (4)$$

where α is the size of the set $T - \{t\}$ and K is either L or R depending on whether the word w_c lies to the left or right of w_e respectively. Notice that this sum is over all the false tags in set T for the word w_e.

With the intuition that the important word should have higher dot product corresponding to true tag than to false tags, we define the score $L_1(w_c, w_e, S)$ as follows:

$$L_1(w_c, w_e, S) = \frac{p_{t,K}^T . h_K - AvgSum(w_c, w_e, S)}{AvgSum(w_c, w_e, S)} \quad (5)$$

We again employ word erasure technique and recompute the above score by replacing the representation of word w_c with a random word representation. We call it $L_2(w_c, w_e, S)$. Now, we can compute the final score for this instance $L(w_c, w_e, S)$ as:

$$L(w_c, w_e, S) = \frac{L_1(w_c, w_e, S) - L_2(w_c, w_e, S)}{L_2(w_c, w_e, S)} \quad (6)$$

The relevance score $I(w_c, e)$ is then computed by taking average of $L(w_c, w_e, S)$ over all instances.

3 Experiments

We consider the task of sequence tagging problem for evaluation and analysis of the proposed methods to interpret neural network models. In particular, we choose the three variants of recurrent neural network models for Named Entity Recognition(NER) task.

3.1 Model architecture

The generic RNN model architecture used for this work is given in figure 1.

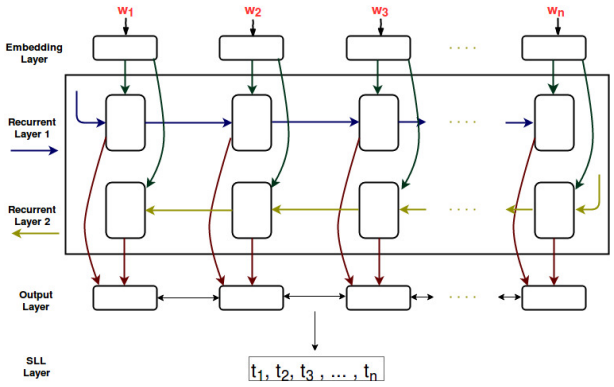


Figure 1: General model architecture for a bi-directional recurrent neural network in sequence tagging problem.

Input layer contains all the words in the sentence. In the embedding layer, each word is represented by it's d dimensional vector representation. The hidden layer contains a bi-directional recurrent neural network which outputs a $2h$ dimensional representation for every word, where h is the number of hidden layer units in the recurrent neural network. In bi-directional models, both the past and future contexts are used to represent the words in a given sentence. Finally, a fully connected network connects the hidden layer to the output layer, which contains scores for each possible tag corresponding to every word in the sen-

tence. A sentence level log likelihood loss function (Collobert et al., 2011) is used in the training process.

For this work, we experiment with standard bi-directional Recurrent Neural Network (Bi-RNN), bi-directional Long Short Term Memory Network (Bi-LSTM) (Graves, 2013; Huang et al., 2015) and bi-directional Gated Recurrent Unit Network(Bi-GRU) (Chung et al., 2014). For simplicity, we refer to these bi-directional models as RNN, LSTM and GRU in rest of the paper.

3.2 Datasets

In this work, we use two NER datasets from diverse domains. One is from generic domain whereas other is from biomedical domain. Statistics of both datasets are given in Table 1.

CoNLL, 2003: This dataset was released as a part of CoNLL-2003 language independent named entity recognition task (Tjong Kim Sang and De Meulder, 2003). Four named entity types have been used: location, person, organization and miscellaneous. For this work, we have used the original split of the English dataset. There were 8 tags used *I-PER*, *B-LOC*, *I-LOC*, *B-ORG*, *I-ORG*, *B-MISC*, *I-MISC* and O. We focus on three entity types, namely, location (LOC), person (PER) and organization (ORG) in our analysis. For this dataset, we use pretrained GloVe 50 dimensional word vectors (Pennington et al., 2014).

JNLPBA, 2004: Released as a part of Bio-Entity recognition task (Kim et al., 2004) at JNLPBA in 2004, this dataset is from GENIA version 3.02 corpus (Kim et al., 2003). There are 5 classes in total - *DNA*, *RNA*, *Cell_line*, *Cell_type* and *Protein*. We use all the classes in our analysis. There are 11 tags, 2 (for begin and intermediate word) for each class and O for other context words. We use 50 dimensional word vectors trained using skip-gram method on a biomedical corpus (Mikolov et al., 2013a; Mikolov et al., 2013b). For this work, we calculate the relevance scores for all the words which have their true tag as O for any test instance in the two datasets.

3.3 Correlation measures

In the output (last) layer we take dot product between weight parameters and the hidden layer outputs and expect that this value (normalized) would be highest corresponding to the true tag. To obtain these similarities between distributions of hidden layer outputs to the weight parameters, we consider two other measures apart from dot product:

1. **Kullback-Leibler Divergence**: Given two discrete probability distributions **A** and **B**, the Kullback-Leibler Divergence(or KL Divergence) from **B** to **A** is computed in the following manner:

$$D_{KL}(A||B) = \sum_i A(i) \log \frac{A(i)}{B(i)} \quad (7)$$

   $D_{KL}(A||B)$ may be interpreted as a measure to see that how good the distribution **B** approximates the distribution **A**. For our experiments, we take normalized weight parameters as **A** and hidden representations as **B**. The lower this KL-divergence is, higher is the correlation between **A** and **B**.

2. **Pearson Correlation Coefficient**: Given two variables **X** and **Y**, Pearson Correlation Coefficient(PCC) is defined as:

$$\rho_{X,Y} = \frac{cov(\mathbf{X}, \mathbf{Y})}{\sigma_X \sigma_Y} \quad (8)$$

   where $cov(\mathbf{X}, \mathbf{Y})$ is the covariance, σ_X and σ_Y are the standard deviations of **X** and **Y** respectively. $\rho_{X,Y}$ takes the values between -1 and 1.

4 Results and Discussion

Throughout our experiments, we use **50** dimensional word vectors, **50** hidden layer units, learning rate as **0.05**, number of epochs as **21** and a batch size of **1**. The performance of various models on both the datasets is summarized in Table 1. Among the three bi-directional models, LSTM performs the best.

4.1 Correlation Analysis

We analyze the correlation between the hidden layer representations and the weight parameters connecting hidden and output layers. Meeting our expectation, this correlation of hidden layer values is found to be higher with the weight parameters corresponding to the true tag for a given input word. For instance, take a sentence from ConLL dataset: "The students, who had staged an 11-hour protest at the junction in northern Rangoon, were taken away in three vehicles.". Here, the word "Rangoon" has it's true tag as *I-LOC* and rest all

Dataset	Instances			Test Set Performance			
	Training	Validation	Testing	Model	Precision	Recall	F Score
CoNLL-2003	14987	3466	3684	RNN	83.42	81.77	82.59
				LSTM	85.87	84.41	**85.13**
				GRU	85.11	83.66	84.38
JNLPBA-2004	18046	500	3856	RNN	67.71	68.99	68.34
				LSTM	67.94	72.69	**70.23**
				GRU	67.55	70.05	68.78

Table 1: Statistics and performance of different models on two NER datasets used in this work.

are context words. Figure 2 plots the normalized values for left side part of the hidden representation for "Rangoon", along with corresponding weight parameters for *I-LOC* and *I-MISC* tags. *I-MISC* has been chosen as it's corresponding dot product is maximum among all the false tags. The high correlation between the hidden representation and weight parameters for the true tag can be clearly observed from the figure.

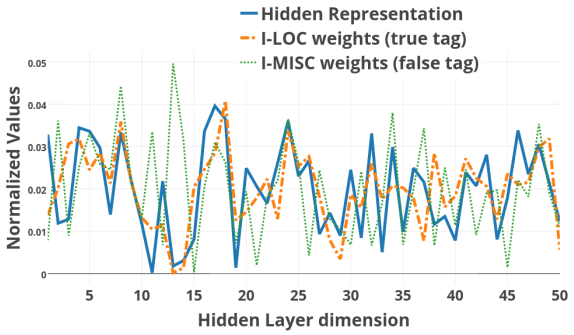


Figure 2: Visualization of hidden representation of a LOC entity word "Rangoon" and weight parameters corresponding to true and false tags.

Table 2 gives the correlation values for above three measures corresponding to the "Rangoon" instance.

Tag	Dot Product	KL Divergence	PCC
I-LOC (True tag)	**7.27**	**0.15**	**0.62**
I-MISC (False Tag)	1.76	0.48	0.17

Table 2: Correlation values obtained corresponding to "Rangoon" instance from CoNLL dataset.

4.2 Analysis of Relevance Scores

In order to evaluate the ability of RNN models to capture important contextual words, we do a qualitative analysis at both word and sentence levels. This section provides instances from both CoNLL and JNLPBA datasets to illustrate how the three measures can be used to identify salient words with respect to bi-directional model. Although we compute word rankings using the three measures described above, our demonstrations in the paper primarily focus on the M_LRC method. M_LRC is able to treat each word individually with due attention to dependency on another words in a given sentence.

At the word level, we further breakdown the visualizations into three types:

Fixing a word and a method: In this case, we fix a particular word and use M_LRC method. We analyze how the importance scores change with various models, entities and correlation measures. Figures 3a, 3b and 3c show heatmaps by fixing the word "midfielder" and M_LRC method for CoNLL dataset. Based on our intuition, the word "midfielder" should have higher importance scores for PER entity. This is clearly visible in the illustrations. All the three correlation measures are able to capture this intuition to a reasonable extent. Similarly, figures 3d, 3e and 3f show heatmaps for "apoptosis" on JNLPBA dataset. The higher scores given to class CT (*cell_type*) are in agreement with the results of M_WF method as well as with our intuition as "apoptosis" indicates cell death. It can also be observed that all the bidirectional models do quite well in both these cases.

Fixing a model and a method: In this case, we fix a particular model and try to visualize how the models score different contextual words for different entity types. Figure 4 shows the heatmaps by fixing RNN, LSTM and GRU respectively with M_LRC method (using dot product). Our intuition that "captain", "city" and "agency" would be relevant for PER, LOC and ORG entities respectively, is proved to be true as can be observed in all of the cases. However, neural models are unable to associate "agency" with ORG as distinctively as in case of "captain" and "city". This can be attributed to frequent occurrence of the word "agency" in the context of words belonging to PER or LOC entities, thereby, confusing the

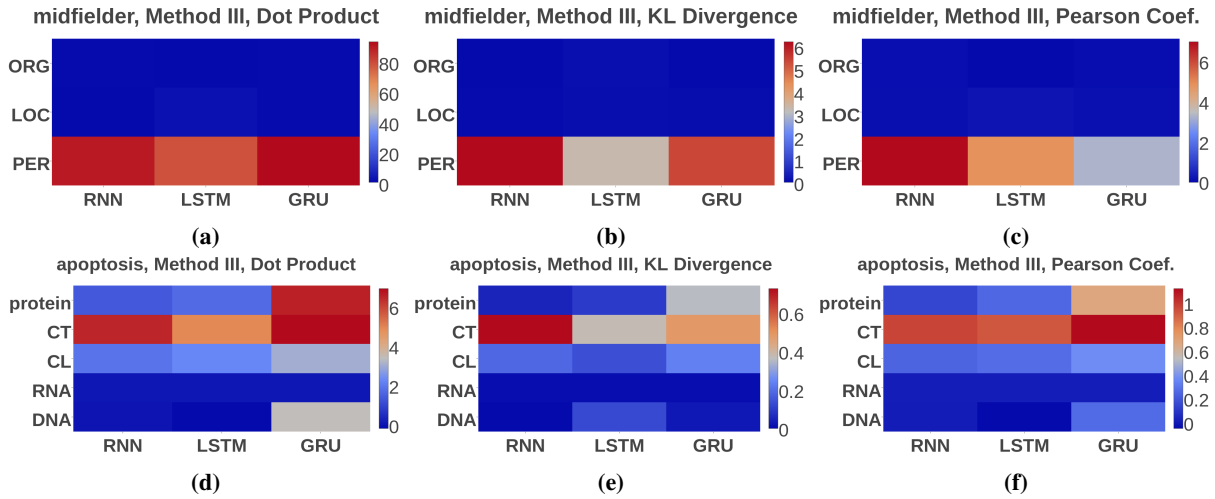


Figure 3: Heatmaps showing the scores for different words across models, entities and methods on CoNLL dataset in part (a), (b) and (c) and on JNLPBA dataset in (d), (e) and (f). Here, CT refers to *cell_type* and CL refers to *cell_line*.

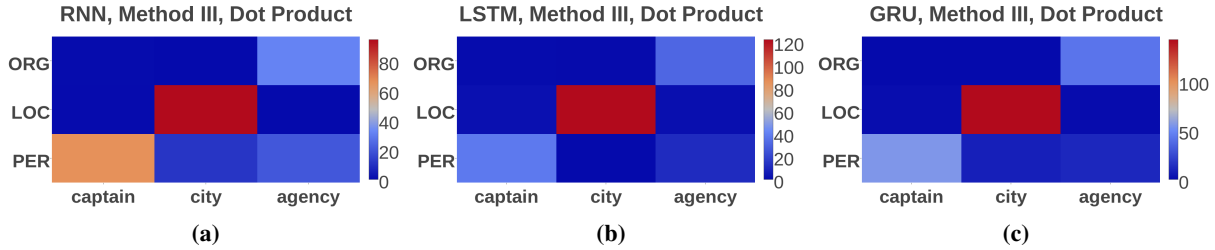


Figure 4: Heatmaps showing the word scores fixing a model with M_LRC method using dot product on CoNLL dataset.

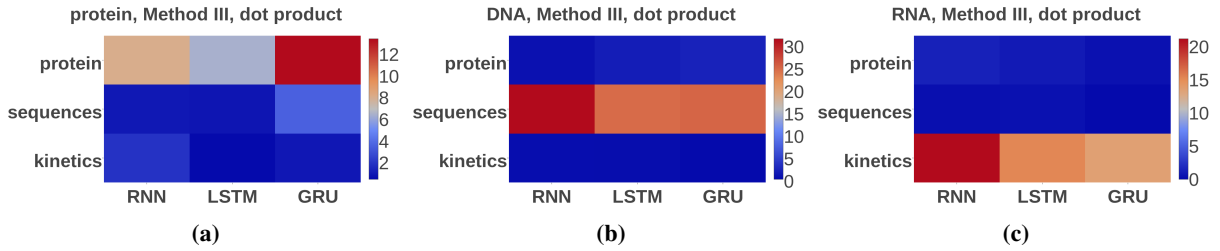
models.

Fixing an entity and a method: Now, we fix a particular entity to analyze which model gives higher importance to different contextual words for a particular entity. Figure 5 shows the heatmaps by fixing entities $protein$, DNA and RNA respectively with M_LRC method. "protein", "sequences" and "kinetics" have high frequency scores for $protein$, DNA and RNA respectively. The models capture this beautifully in all the cases.

At a sentence level, we only consider our best performing model, LSTM. Table 3 gives entity wise word relevance scores for two individual sentences. It uses a sentence from CoNLL dataset - "Saturday 's national congress of the ruling Czech (*I-ORG*) Civic (*I-ORG*) Democratic (*I-ORG*) Party (*I-ORG*) ODS (*I-ORG*)) will discuss making the party more efficient and transparent , Foreign Minister and ODS (*I-ORG*) vice-chairman Josef (*I-PER*) Zieleniec (*I-PER*), said on Friday .". The tags for all entity words are mentioned alongside each word. Notice the high scores for "vice-chairman", "ruling", "congress", "minister" meets the intuitive understanding of these words. Interestingly, round brackets get the maximum scores for M_SLL method, which may be attributed to their frequent use with ORG entity words. Similarly, sentence taken from JNLPBA dataset is: "the number of glucocorticoid (*B-protein*) receptor (*I-protein*) sites in lymphocytes (*B-cell˙type*) and plasma cortisol concentrations were measured in dgdg patients who had recovered from major depressive disorder and dgdg healthy control subjects .". Again, higher scores for "sites" and "plasma" for $cell\_type$ are in agreement with overall scores given to them.

4.3 Positional effects of context words

In this section, we analyze how the position of context words affects their scores obtained by M_LRC method. We do this analysis for real sentences present in the test sets as well as on artificial sentences. We achieve this by applying the proposed techniques at an individual sentence level. For instance, Table 4 shows the relevant scores of the word "minister" for entity PER obtained by three models, in three test sentences taken from CoNLL dataset. M_WF method indicates that "minister" has high importance for en-



(a) **(b)** **(c)**

Figure 5: Heatmaps showing the word scores fixing M_LRC method and entities on JNLPBA dataset.

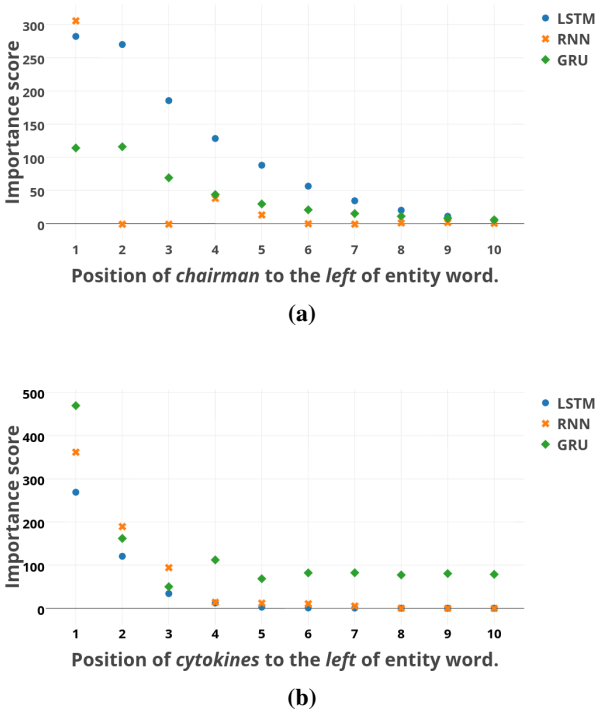
Word	Score
(	9.407
,	8.428
ruling	2.537
vice-chairman	1.41
of	1.203
national	0.901
discuss	0.732
congress	0.728
the	0.723
's	0.486
minister	0.403
and	0.209
saturday	0.065
0	0.03
on	0
friday	0
)	-0.002
said	-0.023
will	-0.045
party	-0.068
making	-0.072
transparent	-0.088
efficient	-0.09
foreign	-0.184
more	-0.202

(a)

Word	Score(Pr)	Score (CT)
control	0	0
and	-0.193	0
major	-0.487	-0.101
number	10.148	2.698
in	0.515	80.745
depressive	7.463	0.039
from	10.221	0.032
had	2.051	0.007
sites	-0.025	18.487
0	0	0
subjects	0	0
plasma	-0.083	0.001
recovered	-0.388	-0.014
cortisol	0.134	0
who	0.933	-0.002
measured	0.639	0.001
healthy	-0.047	0
of	36.08	4.335
dgdg	-0.343	-0.001
patients	3.377	0.007
were	0.454	0.001
concentrations	0.014	0
the	-0.613	2.572
disorder	10.723	0

(b)

Table 3: Entity wise relevance scores for words in two individual sentences using LSTM model: **(a)** Using M_SLL method for CoNLL instance and **(b)** Using M_LRC method with dot product for JNLPBA instance.



(a)

(b)

Figure 6: Position vs relevance score plot for three models for (a) "chairman" w.r.t. PER entity word "Josef" and (b) "cytokines" w.r.t. $protein$ entity word "erythropoietin".

tity type PER matching with our intuition. However "minister" is likely to appear in different sentences with different context and may not have equal relevance as also indicated in the Table 4. In the first sentence, there is no entity word for PER, hence, the score for "minister", corresponding to entity PER is zero. In the second sentence, the score is higher, though not too high as the word is relatively far from the relevant entity word. However, the score is much higher in the third sentence where "minister" is right before the entity words "Margaret Thatcher". Relative scores obtained by using different neural models also match with the general notion that RNN tends to forget long range context (second sentence) compared to LSTM and GRU, and is quite good for short distance context (third sentence).

We further validate the above observation on artificial examples. Figure 6a gives the position verses score plot for the word "chairman" with respect to the PER entity word "Josef". The position tells that how far to the left "chairman" is from the entity word. We create sentences as follows - "chairman Josef .", "chairman R Josef .", "chairman R R Josef ." and so on. Here, R represents a random word. It can be observed that how LSTM and GRU assign a higher score to far off words compared to RNN, justifying their ability to include such words in making the final decision.

Figure 6b shows a similar plot for the word "cytokines" and a $protein$ entity word "erythropoietin" using the same way of creating artificial sentences. Interestingly, GRU assigns higher relevance scores than LSTM and RNN, which is in accordance with the high overall score it gives to

RNN	LSTM	GRU	Sentence
0.0	0.0	0.0	Senegal proposes foreign **minister** for U.N. post .
0.163	2.576	1.031	He was senior private secretary to the employment and industrial relations **minister** from 1983 to 1984 and was Economic advisor to the treasurer Paul Keating in 1983 .
239.793	112.405	199.985	The ODS , a party in which Klaus often tries to emulate the style of former British Prime **Minister** Margaret Thatcher , has been in control of Czech politics since winning general elections in 1992

Table 4: Relevance scores for the word "minister" in three different test sentences from CoNLL dataset.

"cytokines" compared to the other two models.

Rank	Word	Score
1	by	66.162
2	the	22.223
3	in	3.576
4	expression	0.257
5	can	0.222
6	gene	0.221
7	which	0.079
8	over	0.079
9	important	0.003
10	may	0.002
11	establishing	0
12	type	0
13	cell	0
14	0	0
15	specificity	0
16	and	0
17	widening	-0.001
18	range	-0.016
19	recognized	-0.364
20	be	-0.475
21	modulated	-0.534
22	degeneracy	-0.857
23	sequences	-0.917

Table 5: Relevance scores for an individual test sentence from JNLPBA dataset, using LSTM and M_LRC method with dot product.

4.4 Error Analysis

The proposed methods can be effectively used to conduct error analysis on bi-directional recurrent neural network models. For a given sentence, a negative score for a particular word means that the model is able to make a better decision when the word is removed from the sentence. Relevance scores can be used to find out which words confuse the model. Knowing what those words are, is crucial to understanding why the model makes a mistake in a particular instance. For example, Table 5 shows the word importances for the sentence - "the degeneracy in sequences recognized by the otfs (*B-Protein*) may be important in widening the range over which gene expression can be modulated and in establishing cell type specificity ." The LSTM model makes a mistake here by tagging "otfs" with tag *B-DNA*. Words "degeneracy", "sequences", "widening", "recognized" and "modulated" all have a higher overall score for DNA entity class than for $protein$. Hence, the presence of these words in the sentence fool the model into making a wrong decision.

In general, we observe that the presence of words which have high scores for false entity types tend to confuse the model. Position of words also plays a vital role. Words which appear in a far off or a different position than what they generally appear in the training dataset, tend to receive negative or low scores even if they are important. For instance, "minister" mostly appears to the left of an entity word in the training dataset. If, in a test case, it appears to the right, it ends up receiving a low score.

5 Related Work

Various attempts have been made to understand neural models in the context of natural language processing. Research in this direction can be traced back to Elman (1989) which gains insight into connectionist models. This work uses principal component analysis (PCA) to visualize the hidden unit vectors in lower dimensions. Recurrent neural networks have been addressed in recent works such as Karpathy et al. (2015). Instead of a sequence tagging task, they use character level language models as a testbed to study long range dependencies in LSTM networks.

Li et al. (2015) build methods to visualize recurrent neural networks in two settings: sentiment prediction in sentences using models trained on Stanford Sentiment Treebank and sequence-to-sequence models by training an autoencoder on a subset of WMT'14 corpus. In order to quantify a word's salience, they approximate the output score as a linear combination of input features and then make use of first order derivatives. Erasure technique helps us to do away with such assumptions

and find word importances in sequence labeling tasks for individual entities.

Similar to present work, Kádár et al. (2016) analyze word saliency by defining an omission score from the deviations in sentence representations caused by removing words from the sentence. This work, however, targets a different, multi-task GRU framework, learning visual representations of images and a language model simultaneously.

Another closely related work is Li et al. (2016). They use erasure technique to understand the saliency of input dimensions in several sequence labeling and word ontological classification tasks. Same technique is used to find out salient words in sentiment prediction setting. Our work focusing on sequence labeling task has several differences with Li et al. (2016). Firstly, in case of sequence labeling, Li et al. (2016) only focus on feed forward neural networks while our work trains three different recurrent neural networks on general and domain specific datasets. Secondly, their analysis in sequence labeling task is only limited to important input dimensions. Instead, our work focuses on finding salient words which are basic units for most NLP tasks. Lastly, our M_SLL method is an adaptation of their method to find salient words in sentiment prediction task. Unfortunately, for a sequence labeling task, this method is not very suitable. Since it only considers sentence level log likelihood, it makes no distinction between various possible entities such as person or organization. Our M_LRC method, which takes individual word level effects into account, is more suitable.

A significant amount of work has been done in Computer Vision to interpret and visualize neural network models (Simonyan et al., 2013; Mahendran and Vedaldi, 2015; Nguyen et al., 2015; Szegedy et al., 2013; Girshick et al., 2014; Zeiler and Fergus, 2014; Erhan et al., 2009). Attention can also be useful in explaining neural models (Bahdanau et al., 2014; Luong et al., 2015; Sukhbaatar et al., 2015; Rush et al., 2015; Xu and Saenko, 2016).

6 Conclusions and Future Work

In this paper, we propose techniques using word erasure to investigate bi-directional recurrent neural networks for their ability to capture relevant context words. We do a comprehensive analysis of these methods across various bi-directional models on sequence tagging task in generic and biomedical domain. We show how the proposed techniques can be used to understand various aspects of neural networks at a word and sentence level. These methods also allow us to study positional effects of context words and visualize how models like LSTM and GRU are able to incorporate far off words into decision making. They also act as a tool for error analysis in general by detecting words which confuse the model. This work paves the way for further analysis into bi-directional recurrent neural networks, in turn helping to come up with better models in the future. We plan to take our analysis further by including other aspects like character and word level embedding into account.

References

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Chung et al.2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

[Elman1989] Jeffrey L Elman. 1989. Representation and structure in connectionist models. Technical report, DTIC Document.

[Erhan et al.2009] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2009. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341:3.

[Girshick et al.2014] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

[Graves2013] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

[Hong2005] Gumwon Hong. 2005. Relation extraction using support vector machine. In *International Conference on Natural Language Processing*, pages 366–377. Springer.

[Huang et al.2015] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

[Kádár et al.2016] Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2016. Representation of linguistic form and function in recurrent neural networks. *arXiv preprint arXiv:1602.08952*.

[Karpathy et al.2015] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

[Kim et al.2003] J-D Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. 2003. Genia corpusa semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182.

[Kim et al.2004] Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. Association for Computational Linguistics.

[Lafferty et al.2001] John Lafferty, Andrew McCallum, Fernando Pereira, et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

[Li et al.2015] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

[Li et al.2016] Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

[Luong et al.2015] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[Mahendran and Vedaldi2015] Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5188–5196.

[Mikolov et al.2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al.2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Nguyen et al.2015] Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[Rush et al.2015] Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

[Simonyan et al.2013] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

[Sukhbaatar et al.2015] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

[Szegedy et al.2013] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

[Tjong Kim Sang and De Meulder2003] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

[Xu and Saenko2016] Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, pages 451–466. Springer.

[Zeiler and Fergus2014] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Correcting General Purpose ASR Errors using Posteriors

Sunil Kumar Kopparapu
TCS Research and Innovation - Mumbai
SunilKumar.Kopparapu@TCS.Com

C Anantaram
TCS Research and Innovation - Delhi
C.Anantaram@TCS.Com

Abstract

Speech based interfaces have gained popularity because of the advances in automatic speech recognition (ASR) technology, more recently, triggered by the use of deep neural networks for acoustic and language modeling. However, the performance of any general purpose ASR (gpASR) is poor especially for low resource languages and the performance deteriorate further for natural conversational speech. In this paper, we propose a statistical approach to learn the corrections to enable the use of a general purpose ASR for domain specific use. The proposed idea is based on the observation that there are three types of errors that occur in an ASR output, namely, insertion, deletion or substitution of a word or a phoneme. We propose to model these errors statistically and use these posteriors to develop a scheme that is able to repair the ASR output. The proposed system is able to handle ASR errors spread across lexical words also.

1 Introduction

Speech is the most natural mode of communication to query for answers (Kopparapu, 2014). Use of natural language speech to query for information is gaining practical applicability in our day to day activities. It is also believed that in the next ten years there will be a ten times increase in the number of speech interface we will be facing in our day to day life (Fuhrmann et al., 2017). However recognition of natural language speech is not always 100% accurate even when a state of the art ASR engine is employed for a resource rich language. There are several reasons, the significant among them are the mismatch in the train and the test conditions in terms of the acoustic modeling, the accent, the language, the environment etc.

There have been several attempts to improve the speech recognition accuracies (a) by fine tuning, adapting or learning the acoustic models (AM) to handle the train-test mismatch condition (Mohamed et al., 2012); and (b) by configuring the statistical language models (SLM) so that the perplexity of the search space can be reduced (Kombrink et al., 2012). However, these attempts are either far from producing accurate speech to text conversion especially for natural language speech or make the speech recognition engine constrained to perform for a very specific task or domain. Subsequently, there have been several attempts to post process (Ainsworth and Pratt, 1992; Nishizaki and Sekiguchi, 2006; Bassil and Alwani, 2012) the output of the speech recognition engine to identify and correct the erroneous output.

Most work on ASR error detection and correction has focused on using confidence measures, generally called the log-likelihood score, provided by the speech recognition engine; the text with lower confidence is assumed to be incorrect and subjected to correction (Shi, 2008; Zhou et al., 2005). Such confidence based methods are useful only when we have access to the internals of a speech recognition engine built for a specific domain. As mentioned earlier, use of domain-specific engine requires one to rebuild the interface every time the domain is updated, or a new domain is introduced. As mentioned earlier, our focus is to avoid rebuilding the interface each time the domain changes by using an existing ASR. As such our method is specifically a post-ASR system. A post-ASR system provides greater flexibility in terms of absorbing domain variations and adapting the output of ASR in ways that are not possible during training a domain-specific ASR system (Ringger and Allen, 1996). More recently,

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 283–289

there have been attempts to use a general purpose speech recognition engine and then correct the ASR output using bio-inspired evo-devo (Anantaram et al., 2015b; Anantaram et al., 2015a) and statistical techniques (Anantaram and Kopparapu, 2017) based on features extracted from the reference and the ASR output text. The machine learning based system described in (Anantaram and Kopparapu, 2017) is along the lines of (Jeong et al., 2004) but differs in the sense that they use of multiple features for training the Naive Bayes classifier instead of a single feature (syllable count) for training used in (Jeong et al., 2004) in addition to not using manual alignment between the ASR and reference text. In (Twiefel et al., 2014) the authors address the post correction of a general purpose ASR by using (a) the closeness of phoneme depending on the place and manner of articulation to identify the confusability between the actual and the recognized phoneme and (b) using the front-end and the linguist module of Sphinx. However their experiments seem to suggest that their task is that of aligning the gpASR output text with another sentence from a known finite set of sentences.

In this paper, we use a novel approach to repair the errors produced by a gpASR engine. We do not assume, unlike (Twiefel et al., 2014) the availability of reference pre defined sentences. In a very broad sense we try to model the performance of the gpASR engine for a certain environment and domain which is then used to repair the ASR output for all speech utterances coming from the same environment and domain. The rest of the paper is organized as follows. In Section 2 we formulate the problem and describe an approach to model the ASR for a specific domain and environment and in Section 3 we describe the dataset used to evaluate the proposed approach and give some preliminary results. We conclude in Section 4.

2 Problem Formulation

Let

$$\vec{R} = /r_1 r_2 r_3 \cdots r_m/$$

be a spoken sentence consisting of M words that is input to a speech recognition engine. For example,

$$\vec{R} = \begin{cases} \texttt{/who is the accountable} \\ \texttt{person for manufacturing} \\ \texttt{solutions/} \end{cases}$$

Let the general purpose automatic speech recognition (gpASR) output,

$$\vec{O} = \text{"}\mathtt{o_1 o_2 o_3 \cdots o_n}\text{"}$$

consisting of say N words such that $\vec{O} \neq \vec{R}$. For example,

$$\vec{O} = \begin{cases} \texttt{"who is accountable boston} \\ \texttt{for the men affecting} \\ \texttt{solutions"} \end{cases}$$

which was obtained using Kaldi (Povey et al., 2011) with Fisher acoustic models (gpAM) and language models (gpLM) (Kaldi, 2015). Namely,

$$\vec{R} \longrightarrow \underbrace{\boxed{\text{gpASR}}}_{\uparrow} \longrightarrow \vec{O}$$
$$\boxed{\text{gpLM, gpAM}}$$

Observe that M can be $\lessgtr N$. However, there are only three type of operations that are possible to transform $\vec{O}$ to $\vec{R}$, namely, a word in $\vec{O}$ is either deleted or is inserted or is substituted so that $\vec{O}$ can become identical to $\vec{R}$. Clearly, one insertion, two deletions and two substitutions to $\vec{O}$ could make it identical to $\vec{R}$, namely,

$$\begin{aligned} &\texttt{"who is (}\phi \overset{ins}{\rightarrow}\texttt{the) accountable} \\ &\texttt{(boston}\overset{sub}{\rightarrow}\texttt{person) for} \\ &\texttt{(the} \overset{del}{\rightarrow} \phi\texttt{) (men} \overset{del}{\rightarrow} \phi\texttt{)} \\ &\texttt{(affecting}\overset{sub}{\rightarrow}\texttt{manufacturing)} \\ &\texttt{solutions"} \end{aligned}$$

$$\vec{O}_p \longleftarrow \underbrace{\boxed{\text{word2phoneme}}}_{\uparrow} \longleftarrow \vec{O}$$
$$\boxed{\text{gpLexicon}}$$

Let each word in $\vec{R}$ and $\vec{O}$ be represented by its phonetic equivalent so that

$$\vec{R}_p = r_1 r_2 \cdots r_M$$

and

$$\vec{O}_p = o_1 o_2 o_3 \cdots o_N$$

such that $\{r_i, o_i\} \in I\!P$ where $I\!P$ is the set of phonemes. Note that r_1 in $\vec{R}_p$ is a phoneme while r_1 in $\vec{R}$ is a lexical word. In general there are 39 phones in $I\!P$, namely,

$$I\!P = \left\{ \begin{array}{l} \texttt{a, ae, ah, ao, aw, ay, b, ch,} \\ \texttt{d, dh, eh, er, ey, f, g, hh, ih,} \\ \texttt{iy, jh, k, l, m, n, ng, ow, oy,} \\ \texttt{p, r, s, sh, t, th, uh, uw, v,} \\ \texttt{w, y, z, zh} \end{array} \right\}$$

Subsequently, we can write

$$\vec{O}_p = \begin{cases} \texttt{hh uw ih z dh iy ah k aw n t ah b} \\ \texttt{ah l p er s ah n f r er m ae n y} \\ \texttt{ah f ae k ch er ih ng s ah l uw sh} \\ \texttt{ah n z} \end{cases} \quad (1)$$

and

$$\vec{R}_p = \begin{cases} \texttt{hh uw ih z ah k aw n t ah b ah l b} \\ \texttt{ao s t ah n f r er dh iy m eh n ah} \\ \texttt{f eh k t ih ng s ah l uw sh ah n z}. \end{cases} \quad (2)$$

Notice that even as a phonemic string, $\vec{O}_p$ can be transformed into $\vec{R}_p$ through one of the three operations, namely, deletion, insertion or substitution.

2.1 Computing Posteriors

We define an extension $\mathbb{P}' = \{\mathbb{P}, \phi\}$, where the element ϕ represents a null-phoneme. Given a corpus of $\{\vec{O}_p^i, \vec{R}_p^i\}_{i=1}^K$ pairs (K is large). Note that the elements of $\vec{O}_p$ and elements of $\vec{R}_p \in \mathbb{P}$ and can take one of the 39 unique phones. Represent o and r as $p \in \mathbb{P}$. Now we compute for all $p_i \in \mathbb{P}$

$$P_{sub} = P(p_i \overset{sub}{\rightarrow} p_j) = \frac{\#((p_i \in \vec{O}_p)\&(p_j \in \vec{R}_p))}{\#(p_i \in \vec{O}_p)} \quad (3)$$

where (a) $\#(p_i \in \vec{O}_p)$ is the count of the phone p_i occurring in $\{\vec{O}_p^i\}_{i=1}^K$ and (b) $\#((p_i \in \vec{O}_p)\&(p_j \in \vec{R}_p))$ is the count of the phone p_i which occurs in $\{\vec{O}_p^i\}_{i=1}^K$ when p_j occurs in $\{\vec{R}_p^i\}_{i=1}^K$. Similarly, we find

$$P_{ins} = P(\phi \overset{sub}{\rightarrow} p_j) \quad (4)$$

where p_j occurs in $\vec{R}_p$ but does not occur in $\vec{O}_p$ and

$$P_{del} = P(p_i \overset{sub}{\rightarrow} \phi) \quad (5)$$

where p_i occurs in $\vec{O}_p$ but does not occur in $\vec{R}_p$. Note that we can compute $P_*(p_i \overset{sub}{\rightarrow} p_j)$ for all $p_i, p_j \in \mathbb{P}'$, clearly P_* is a 40×40 matrix. The last column corresponds to P_{del} while the last row corresponds to P_{ins}. We conjecture that these posterior probabilities $P_*(p_i \overset{sub}{\rightarrow} p_j)$ can be used to model the ASR engine (see Figure 1) which in turn can be used to repair the ASR output, namely,

$$\vec{O}_p \longrightarrow \underbrace{\boxed{\text{repair}}}_{\uparrow P_*} \longrightarrow \vec{R}_p$$

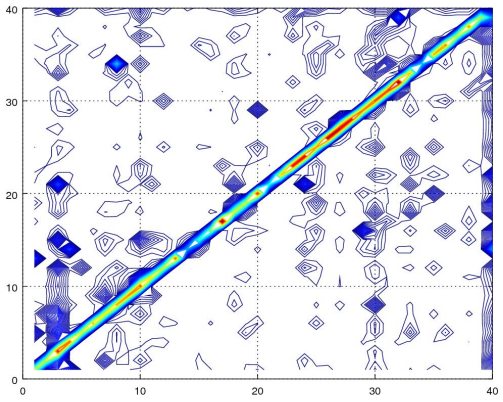


Figure 1: P_* for Kaldi (Povey et al., 2011) with Fischer acoustic and language models. Note that there are higher P_{ins}.

2.2 Repair Approach

Now given an general purpose ASR output, say, $O_1O_2O_3O_4O_5 \cdots O_N$ identify the words O_i which are not part of the domain along the lines of (Anantaram et al., 2015a), using a domain ontology. The domain ontology (dLexicon) helps in identifying all the O_i's such that $O_i \notin$ dLexicon.

Then using the domain lexicon we construct for each O_i the phoneme sequence $o_{i1}, o_{i2}, o_{i3}, o_{i4}, o_{i5}, \cdots, o_{im}$ where each $o_{ij} \in \mathbb{P}$. We then expand the phonemes that require correction as

$$\begin{aligned} expand(o_i, \phi) \quad = \quad & \phi, o_{i1}, \phi, o_{i2}\phi, o_{i3}, \phi, o_{i4}, \\ & \phi, o_{i5}, \phi, \cdots, \phi, o_{im}, \phi \end{aligned} \quad (6)$$

by inserting ϕ between the phonemic representation of O_i. This extended phoneme string is corrected using the posterior (P_*). Now for all identified o_{ij} and $\phi \in \epsilon$, find all $p_k \in \mathbb{P}'$ such that the posterior

$$P_*(o_j \overset{sub}{\rightarrow} p_k) > \tau \text{ and } P_*(\phi \overset{sub}{\rightarrow} p_k) > \tau. \quad (7)$$

We form a lattice ($\mathcal{L}$) using p_k obtained in (7) , namely,

$$\epsilon \longrightarrow \boxed{P_*} \longrightarrow \mathcal{L}$$

We search through this lattice, $\mathcal{L}$, to identify the best set of p_k that results in the highest probability score such that the resulting phoneme string represents a word in the domain, namely,

$$\mathcal{L} \longrightarrow \underbrace{\boxed{\text{search, phoneme2word}}}_{\uparrow \boxed{\text{dLexicon}}} \longrightarrow O'$$

where O' is the corrected word corresponding to O_i. Algorithm 1 captures this approach in greater details.

Algorithm 1 Repair $\vec{O}$.

Given $\vec{O} = O_1 O_2 O_3 O_4 O_5 \cdots O_N$
Given τ, P_*/* Posterior */
Given $cnt = 0$, dLexicon /* Domain Ontology */

for $k = 1, 2 \cdots N$ **do**
  if $O_k \notin$ dLexicon **then**
    $\alpha[cnt++] \longleftarrow O_k$
    /* α contains all words $\notin$ dLexicon */
  end if
end for

for $l = 1, 2, \cdots, cnt$ **do**
  $\vec{o} \leftarrow O_l$ /* Using gpLexicon */
  $\vec{o}_l \leftarrow expand(\vec{o}, \phi)$ /* Expand using (6)*/
  for $p_j \in \vec{o}_l$ **do**
    $knt = 0$
    for $(p_k \in I\!P')$ **do**
      if $(P(p_j \overset{sub}{\rightarrow} p_k) > \tau)$ **then**
        $\mathcal{L}_{j,knt++} = \left\{p_k, P(p_j \overset{sub}{\rightarrow} p_k)\right\}$
      end if
    end for
    /* $\mathcal{L}$ is the lattice */
  end for

  Search through $\mathcal{L}$ to find the repaired $\vec{o}\,'_l$
end for

$O'_l \leftarrow \vec{o}\,'_l$ /* Using Lexicon */
/* O'_l is the corrected word */

3 Experimental Results

3.1 Data setup

Experiments were carried out on a database of 700 spoken utterances by 7 different people, each of them speaking 100 sentences. Each of the 7 speakers were given a set of 100 different queries which they were required to speak into a data collection application built in-house. Each of them spoke 10 queries in a session and the 10 sessions were recorded over a period of one or two days. The spoken utterance was recorded in the wave format with 16 bits resolution and a sampling rate of 16 kHz. These 700 spoken utterances were first converted to text using a general purpose speech recognition engine, in our case Kaldi (Povey et al., 2011) using the Fischer acoustic (Kaldi, 2015) and Fischer language models. We built the gpLexicon using the 700 decoded text outputs using an online tool (CMU, 2017). Note that the general purpose ASR (Kaldi) can output words which need not always be part of the domain Ontology (in our case the domain was related to software industry). In all there were 1426 unique words in the 700 ASR text output (gpLexicon) and there were 372 unique words in the domain lexicon (dLexicon). We divided the 700 into 5 sets of 140 utterances each and carried out a 5 fold validation, namely we used 4 sets, consisting of 560 sentences to compute the posterior and 1 set consisting of 140 sentences to test. In this paper, we present a sample example to demonstrate the effectiveness of our approach.

3.2 Construction of P_*

We constructed the posteriors using the decoded text (example, (1)) and the actual spoken text (example, (2)) after converting both the text strings into phonemes using a phonetic lexicon (we used gpLexicon for ASR output text and dLexicon for the actual spoken text). We aligned the two phonetic strings using edit distance algorithm which came with (Povey et al., 2011). For (1) and (2) we get

"hh, uw, ih, z, dh, (ae $\overset{sub}{\rightarrow}$ ah), (t $\overset{del}{\rightarrow}$ ϕ), ah, k, aw, n, t, ah, b, ah, l, (b $\overset{sub}{\rightarrow}$ p), (aa $\overset{sub}{\rightarrow}$ er), s, (t $\overset{del}{\rightarrow}$ ϕ), ah, n, f, ao, r, (dh $\overset{del}{\rightarrow}$ ϕ), (ah $\overset{del}{\rightarrow}$ ϕ), m, (eh $\overset{sub}{\rightarrow}$ ae), n, (ϕ $\overset{ins}{\rightarrow}$ y), ah, f, (eh $\overset{sub}{\rightarrow}$ ae), k, (t $\overset{sub}{\rightarrow}$ ch), (ih $\overset{sub}{\rightarrow}$ er), (ϕ $\overset{ins}{\rightarrow}$ ih), ng, s, ah, l, uw, sh, ah, n, z"

which are used to compute the posterior as mentioned earlier. As seen there are 7 substitutions, 3 deletions and 2 insertions. A sample posterior P_* is shown in Figure 1 as a contour plot. The x-axis represents the phonemes (from the actual text) while the y-axis is the phonemes (in the words recognized by gpASR).

3.3 Sample Repair

As an example the general purpose ASR returned

```
"who is that accountable
boston for the men affecting
solutions"
```

when

```
/who is the accountable
person for manufacturing
solutions/
```

	"boston"												
		b		ao		s		t		ah		n	
1	$\boldsymbol{\phi}$	b	$\boldsymbol{\phi}$	ao	$\boldsymbol{\phi}$	**s**	$\boldsymbol{\phi}$	t	$\boldsymbol{\phi}$	**ah**	$\boldsymbol{\phi}$	**n**	$\boldsymbol{\phi}$
2	ah	**p**	ah	ah	ah	sh	ah	$\boldsymbol{\phi}$	ah	$\boldsymbol{\phi}$	ah	ϕ	ah
3	ih	ϕ	ih	ϕ	ih	ϕ	ih	r	ih	er	ih	l	ih
4	k	r	k	**er**	k	z	k	ah	k	r	k	ah	k
5	t	w	t	aa	t	ah	t	er	t	aa	t	k	t
6	l	l	l	ow	l	er	l	l	l	ae	l	r	l
7	r	ah	r	iy	r	ey	r	d	r	ih	r	ae	r
8	s	dh	s	sh	s	r	s	th	s	eh	s	ih	s
9	d	d	d	p	d	ih	d	dh	d	ey	d	m	d
10	w	t	w	v	w	ch	w	eh	w	uw	w	ng	w
		p		**er**		**s**				**ah**		**n**	
	"person"												

Table 1: Lattice $\mathcal{L}$ constructed using the posterior P_*, used to correct 'b ao s t ah n' to 'p er s ah n'. The phones marked in bold are the ones that are picked during the lattice search.

was given as the audio input to the gpASR. The output text "boston" and "men affecting" are not part of the domain ontology (dLexicon).

We first converted the word "boston" into its phoneme equivalent using the general purpose phonetic lexicon (gpLexicon), namely 'b ao s t ah n', then using the posterior P_* we constructed the lattice $\mathcal{L}$ (see Table 1) which shows the top 10 phonemes that could replace the phoneme output by the ASR based on the posterior P_*. For example, the column associated with b shows that $P(\texttt{b} \overset{sub}{\rightarrow} \texttt{p}) \geq P(\texttt{b} \overset{del}{\rightarrow} \phi) \geq P(\texttt{b} \overset{sub}{\rightarrow} \texttt{r})$ etc. Note that we have captured the possible insertion by appending ϕ between the phones, namely, ϕ b ϕ ao ϕ s ϕ t ϕ ah ϕ n (6). Now traversing the lattice $\mathcal{L}$ in Table 1 from left to right we can get $\boldsymbol{\phi}$ **p** $\boldsymbol{\phi}$ **er** $\boldsymbol{\phi}$ **s** $\boldsymbol{\phi}$ $\boldsymbol{\phi}$ $\boldsymbol{\phi}$ **ah** $\boldsymbol{\phi}$ **n** $\boldsymbol{\phi}$ which is nothing but 'p er s ah n' which is part of our domain.

A similar repair mechanism described in Algorithm 1 helps in correcting "men affecting" to the domain word "manufacturing" as shown in Figure 2. As a result the general purpose ASR output

> "who is that accountable boston for the men affecting solutions"

is corrected to

> "who is that accountable person for the manufacturing solutions"

Note that this is not exactly what was spoken, namely,

> */who is the accountable person for manufacturing solutions/*

Notice that the repair mechanism is unable to correct "that" to "the" and delete "the" however the repair mechanism is able to correct the non-domain words by identifying phonetically equivalent words in the domain. In this example and all our experiments we choose τ in a manner that we had the top 10 phonemes to form the lattice $\mathcal{L}$.

Also notice that because the repair mechanism operates in the phoneme space it is able to operate even if the error is spread across words (example, "men affecting" $\longrightarrow$ "manufacturing"). This example sentence clearly demonstrates the ability of the proposed repair approach to correct the output of a general purpose automatic speech recognition engine based on the computed posteriors.

We measured the accuracy of improvement in the ASR output, by looking at the edit distance between ($\vec{O}$, $\vec{R}$) and the edit distance between ($\vec{O}'$, $\vec{R}$). Here $\vec{R}$ is the reference out (perfect ASR output), while $\vec{O}$ is the output of the general purpose ASR (in our case Kaldi ASR output) and $\vec{O}'$ is the repaired output based on Algorithm 1. It is observed that the $dis(\vec{O}', \vec{R}) \leq dis(\vec{O}, \vec{R})$. Namely, the repaired output was always closer to the reference that the output of the general purpose ASR engine ($\vec{O}$).

"men"							"affecting"													
	m		eh		n		ah		f		eh		k		t		ih		ng	
ϕ	**m**	ϕ	eh	ϕ	**n**	ϕ	**ah**	ϕ	**f**	ϕ	eh	ϕ	**k**	ϕ	t	ϕ	**ih**	ϕ	**ng**	ϕ
ah	ϕ	ah	ah	ah	ϕ	ah	ϕ	ah	v	ah	ah	ah	ϕ	ah	ϕ	ah	ey	ah	n	ah
ih	n	ih	**ae**	ih	l	ih	er	ih	ah	ih	**ae**	ih	t	ih	r	ih	ah	ih	ϕ	ih
k	l	k	ϕ	k	ah	k	r	k	ϕ	k	ϕ	k	ah	k	ah	k	eh	k	d	k
t	ih	t	ih	t	k	t	aa	t	p	t	ih	t	f	t	**er**	t	ϕ	t	v	t
	m		**ae**		**n**		**ah**		**f**		**ae**		**k**		**er**		**ih**		**ng**	
"manufacturing"																				

Table 2: Lattice $\mathcal{L}$ constructed using the posterior P_*, used to correct "men affecting".

4 Conclusions

Speech based applications are being increasingly deployed for self help in enterprises. However for resource deficient languages (including Indian English) the performance of ASR engine is poor especially for natural spoken utterances. While there are two ways of getting over the poor performance of ASR engine, namely (a) fine tuning the AR engine in terms of acoustic and language models; thereby making the ASR engine domain specific or (b) using a general purpose ASR engine and then correcting the ASR output using domain ontology and in some way modeling the ASR behavior. The advantage of the second approach is that of being able to use a readily available state of the art ASR engine without having to build a unique speech recognition engine for every application. In this paper, we approach the problem of correcting the general purpose ASR output using posteriors. The main contribution of this paper is the formulation of a posterior approach to repair the output of a general purpose ASR engine as depicted in detail in Algorithm 1. We also showed that the approach is able to repair the errors that might be spread across lexical words.

5 Acknowledgements

The authors would like to thank Chirag Patel who assisted in performing some experiments in the earlier stages of this work.

References

W.A. Ainsworth and S.R. Pratt. 1992. Feedback strategies for error correction in speech recognition systems. *International Journal of Man-Machine Studies*, 36(6):833 – 842.

C. Anantaram and Sunil Kumar Kopparapu. 2017. Adapting general-purpose speech recognition engine output for domain-specific natural language question answering. *CoRR*, abs/1710.06923.

C. Anantaram, Rishabh Gupta, Nikhil Kini, and Sunil Kumar Kopparapu. 2015a. Adapting general-purpose speech recognition engine output for domain-specific natural language question answering. In *Workshop on Replicability and Reproducibility in Natural Language Processing: adaptive methods, resources and software at IJCAI 2015*, Buenous Aires.

C Anantaram, Nikhil Kini, Chirag Patel, and Sunil Kopparapu. 2015b. Improving ASR recognized speech output for effective NLP. In *The Ninth International Conference on Digital Society ICDS 2015*, pages 17–21, Lisbon, Portugal.

Youssef Bassil and Mohammad Alwani. 2012. Post-editing error correction algorithm for speech recognition using bing spelling suggestion. *CoRR*, abs/1203.5255.

CMU. 2017. The CMU pronouncing dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

Ferdinand Fuhrmann, Anna Maly, Christina Leitner, and Franz Graf. 2017. Three experiments on the application of automatic speech recognition in industrial environments. In Nathalie Camelin, Yannick Estève, and Carlos Martín-Vide, editors, *Statistical Language and Speech Processing - 5th International Conference, SLSP 2017, Le Mans, France, October 23-25, 2017, Proceedings*, volume 10583 of *Lecture Notes in Computer Science*, pages 109–118. Springer.

Minwoo Jeong, Byeongchang Kim, and G Lee. 2004. Using higher-level linguistic knowledge for speech recognition error correction in a spoken q/a dialog. In *HLT-NAACL special workshop on Higher-Level Linguistic Information for Speech Processing*, pages 48–55.

Kaldi. 2015. Kaldi fisher english. http://kaldi-asr.org/downloads/build/2/sandbox/online/egs/fisher_english/.

Stefan Kombrink, Tomas Mikolov, Martin Karafiát, and Lukás Burget. 2012. Improving language models for ASR using translated in-domain data. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012*, pages 4405–4408. IEEE.

S.K. Kopparapu. 2014. *Non-Linguistic Analysis of Call Center Conversations*. SpringerBriefs in Electrical and Computer Engineering. Springer International Publishing.

A. Mohamed, G.E. Dahl, and G. Hinton. 2012. Acoustic modeling using deep belief networks. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):14 –22, jan.

Hiromitsu Nishizaki and Yoshihiro Sekiguchi. 2006. Word error correction of continuous speech recognition using web documents for spoken document indexing. In Yuji Matsumoto, RichardW. Sproat, Kam-Fai Wong, and Min Zhang, editors, *Computer Processing of Oriental Languages. Beyond the Orient: The Research Challenges Ahead*, volume 4285 of *Lecture Notes in Computer Science*, pages 213–221. Springer Berlin Heidelberg.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, number Idiap-RR-04-2012, Rue Marconi 19, Martigny, December. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

E.K. Ringger and J.F. Allen. 1996. Error correction via a post-processor for continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 427–430 vol. 1, May.

Yongmei Shi. 2008. *An Investigation of Linguistic Information for Speech Recognition Error Detection*. Ph.D. thesis, University of Maryland, Baltimore County, October.

Johannes Twiefel, Timo Baumann, Stefan Heinrich, and Stefan Wermter. 2014. Improving domain-independent cloud-based speech recognition with domain-dependent phonetic post-processing. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1529–1535. AAAI Press.

Lina Zhou, Jinjuan Feng, A. Sears, and Yongmei Shi. 2005. Applying the naive bayes classifier to assist users in detecting speech recognition errors. In *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 183b–183b, Jan.

Retrieving Similar Lyrics for Music Recommendation System

Braja Gopal Patra[1], Dipankar Das[2], and **Sivaji Bandyopadhyay[2]**
[1]School of Biomedical Informatics,
The University of Texas Health Science Center at Houston, Houston, Texas, USA
[2]Department of Computer Science & Engineering, Jadavpur University, Kolkata, India
brajagopal.cse@gmail.com, dipankar.dipnil2005@gmail.com,
sivaji_cse_ju@yahoo.com

Abstract

Presently, millions of music tracks are available on the web. Therefore, a music recommendation system can be helpful to filter and organize music tracks according to the need of users. To develop a recommendation system, we need an enormous amount of data along with the user preference information. However, there is a scarcity of such dataset for Western as well as Hindi songs. This paper presents a similar lyrics retrieval system for Hindi songs using features collected from lyrics. A romanized Hindi lyric dataset is collected from the web. The collected dataset is noisy, and several forms of a single word are present in it, thus an unsupervised stemming algorithm is proposed to reduce the size of N-grams. The Self-Organizing Feature Maps (SOFMs) based similar lyrics retrieval system achieves the maximum F-measure of 0.749.

1 Introduction

The improvement in digital technology has led the music digitally available to all Internet users. The development of nanotechnology made storage devices portable, and nowadays, any handheld devices can store thousands of tracks. Whenever a user has an enormous number of choices for listening to music (like browsing web or personal storage devices), the user is overwhelmed by options. The recommender system comes as a savior and filters the songs that are suitable for that user at that moment. It also maximizes the user's satisfaction by playing appropriate song at the right time, and, meanwhile, minimize the user's effort (Hu, 2014). The recommendation problem can be seen as a ranking problem, and it creates a list of suitable songs for users.

Many music streaming services for Western music emerged in recent years, such as *Google Play Music*[1], *Apple music*[2], *Last.fm*[3], *Pandora*[4], *Spotify*[5], and so forth; and some of them are not available in India. These music streaming applications store user preferences and recommend users what they want to listen. In India, several music streaming services were started recently and those are *Apple music*, *Gaana*[6], *Hungama*[7], *Saavn*[8], and *Wynk music*[9] etc. Most of them do not recommend songs and those are just a music library. Youtube is one of the video streaming services which provides recommendations based on the collaborative filtering (Davidson et al., 2010). It also provides facility to search using title of song and it can also search a video using any keywords within a lyric body only when full lyric is available in the description.

There is a keen interest in accessing music contents nowadays. Available search engines or information retrieval (IR) systems allow users to search a song by the metadata such as song title, artist, album name. Incorrect metadata can lead to wrongly searched data, and without any metadata, it is not possible to search a song. Again, the current IR systems give particular search results based on the query rather than similar lyrics to a query. It was observed that a lyrics provide different semantic information than audio for some of Hindi songs, i.e., the annotators perceived differ-

[1]https://play.google.com/music/listen
[2]https://www.apple.com/music
[3]https://www.last.fm
[4]https://www.pandora.com
[5]https://www.spotify.com
[6]https://gaana.com
[7]http://www.hungama.com
[8]https://www.saavn.com
[9]https://www.wynk.in/music

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 290–297

ent moods while reading lyrics and listening to the corresponding songs (Patra et al., 2016b). People are interested in listening to songs specific to situation and mood (Duncan and Fox, 2005). There is a need for recommendation system based on information within the music as well as the metadata of music such as mood, genre, artist name, and so on.

Music similarity measures can help to understand why two music pieces are perceived alike by the listener and to guide the user in efficiently retrieving desired piece of music (Schedl et al., 2011). Query by hamming helps to find an exact song with respect to a query humming. Again, a lyrics based retrieval system could be helpful for searching similar songs. Till date, there is no such lyrics retrieval system developed for Hindi songs.

In the present task, we collected a huge lyric dataset for Hindi songs written in Romanized English characters. Though, developing lyrics retrieval system for Hindi songs of *Romanized* characters is a difficult task. The main reason is that the processing of such text is difficult for an n-gram based system as a single word is written with different variations, for example, "*Ajnabi*" and "*Ajnabiii*". The existing stemming and parts-of-speech taggers are available for either *utf* or *WX* format of Hindi characters. All sentiment lexicons are also available in *utf* format and these can not be used for *Romanized* characters.

Several text normalization techniques and an unsupervised stemming algorithm have been implemented to handle unstructured data. Finally, we developed unsupervised IR systems to retrieve similar songs with respect to a query song using Self-Organizing Feature Maps (SOFMs) and Document level word embeddings followed by a baseline system using Fuzzy C-means (FCMs) clustering. The similar lyrics retrieval system can be combined with existing metadata based recommender to give a better performance. It is also useful for recommending a song where little or no metadata (genre, mood) is available.

This paper is organized as follows: Section 2 describes the related work on similar lyrics retrieval and works on Indian music. The dataset and preprocessing techniques are discussed in Section 3. Section 4 describes SOFMs and Doc2Vec for developing the retrieval system. The developed systems with comparisons are described in Section 5. Finally, Section 6 concludes and provides avenues for further work.

2 Related Work

Automatic playlist generation is one of the fundamental problem in music information retrieval (MIR) to overcome the manual song selection. Automatic playlist generation can be seen as recommendation problem. The biggest challenges faced while developing recommendation system are collecting a huge dataset and metadata, then getting user preferences or feedback. The recommender system can be developed based on both audio and lyrics to solve the problem of manual playlist selection or generation.

There have been multiple experiments which process lyrics. Mahedero et al. (2005) performed the language identification, structure extraction, theme extraction, and similarity searches mainly on Western lyrics. The mood (Hu et al., 2009; Zaanen and Kanters, 2010) and genre (Mayer et al., 2008) classification have also been performed using lyric features of Western music.

Another interesting task named as LYRICSRADAR was developed by Sasaki et al. (2014) and they visualized the topics of Japaneses lyrics by using a Latent Dirichlet Allocation (LDA). Several experiments were performed on retrieving similar lyrics for Western songs by (Mahedero et al., 2005; Knees et al., 2007; Schedl et al., 2011), Mandarin lyrics by (Wang et al., 2010), and Chinese lyrics by (Han et al., 2015).

2.1 Experiments on Indian Songs

MIR in Indian songs is at early stage. Recently, mood classification of Hindi songs have been performed using audio (Ujlambkar and Attar, 2012; Patra et al., 2013; Patra et al., 2016a), lyrics (Patra et al., 2015), and combination of both (Patra et al., 2016b; Patra et al., 2016c). The datasets used in above experiments are small and not adequate for development of recommendation system.

Some other tasks like raga identification of south Indian Carnatic music (Sridhar et al., 2011), multimodal sentiment analysis of Telugu songs (Abburi et al., 2016), melody identification of Carnatic music (Koduri et al., 2011), rhythm analysis of Indian art music (Srinivasamurthy et al., 2014) etc. have been performed till date. To the best of author's knowledge, almost no work exists for retrieving similar lyrics for any of the Indian songs.

3 Dataset and Preprocessing

3.1 Dataset

As no task on retrieving the similar lyrics has been performed till date, no dataset is also available. A total of 31,171 lyrics of Hindi songs have been collected from several websites[10][11][12] using a web crawler developed by us. Among them, 25,088 lyrics are in *Romanized* characters and 6,083 lyrics are in *utf-8* characters. It is good to develop similar lyrics retrieval system on the lyrics having *utf-8* characters, but the number of such lyrics is insufficient to develop an IR system. Thus, we discarded the latter set of lyrics for the current experiment and developed similar lyrics retrieval system only on the *Romanized* lyrics.

There were many HTML tags and other junk characters, thus, several preprocessing steps were performed on the collected dataset to ensure the quality. The variations in *Romanized* words motivated us to remove the duplicate characters and perform stemming.

3.2 Preprocessing

We removed HTML tags and junk characters from the lyrics. *Mukhda* (the starting stanzas of a song) is repeated in lyric and the importance of words in *mukhda* is higher than the words in *antara* (inside stanzas of a lyric) (Beaster-Jones and Sarrazin, 2016). Again, we observed that the systems are biased towards *mukhda* because of the higher word frequency. We performed our experiments both before and after removing repeated lines from the lyrics.

As the number of n-grams is quite high, and a huge computational power is required to perform the search. Thus, preprocessing is an important step reduce the n-gram size and the steps for preprocessing are sequentially discussed as follows.

3.2.1 Removing Duplicate Characters

There were many words having multiple repeated characters. To reduce the n-gram size, the frequency of any repetitive character were reduced to two. For example, the word '*Ajnabiiiiiiiii*' contains multiple '*i*' and those multiple occurrences of '*i*' was replaced by '*ii*'. At the end, word '*Ajnabiiiiiiiii*' became '*Ajnabii*'. Later on, the proposed stemming algorithm is used on above word to reduce *ii* to *i*.

3.2.2 Stemming Algorithm

It was observed that stemming algorithm improves the performance of any information retrieval system (Moral et al., 2014). Many words are written in different forms, for example, the word '*marega*' (to beat) is written as '*maregaa*' in another lyric. There are several tool for stemming or lemmatization, but all of them are for either *WX* or *utf-8* characters and these tools are not useful for current scenario. Thus, an unsupervised stemming algorithm was developed to reduce the number of n-grams present in corpus and to improve the system performance. We hope that the proposed unsupervised stemming algorithm is useful for handling noisy data from different languages.

The algorithm contains two main steps namely collecting suffix and stemming. The first step describes how suffixes are collected from words by comparing the similar words. Details of the first step is given below.

First, all unique words are stored in a dictionary after sorting them alphabetically and length wise. This step is performed to reduce the number of matching during stemming. For each word, a suffix is searched by comparing with another word starting with same character. If the difference between two words are less than equals to three then rest of the words (after removing the common characters) is considered as a suffix and the word matching is done from the left to right. For example, the words *Ajnabi* and *Ajnabii* have only single character difference, i.e. *i*. Thus, *i* is considered as suffix and inserted in the suffix list. If difference between the words is more than 3, then rest of the words after removing common characters may be a probable suffix. Such suffixes are collected and checked manually before implementing. The second step describes how stemming is performed using the collected suffixes and all inflected words are removed from the final dictionary. The pseudo code for normalizing the words is given in algorithm 1.

After using the stemming algorithm, the word '*maregaa*' is normalized to '*marega*'. Several words having different suffixes at the end were observed in lyrics, for example, '*dost*' (friend), '*dosti*' (friendship), '*doston*' (friends) and the words '*dosti*' and '*doston*' are normalized to '*dost*'.

We removed stopwords while constructing the

[10] http://www.lyricsmint.com
[11] http://smriti.com
[12] http://www.indicine.com

Algorithm 1 Pseudo code for unsupervised stemming

procedure COLLECTING SUFFIX
    Store all unique words in $dict$ after sorting them alphabetically and length wise
    for each $word_i$ in $dict$ **do**
        for each $word_j$ in $dict$ **do**
            if $word_j$.startswith($word_i$) and len($word_j$)-len($word_i$) ≤ 3 **then**
                $word_j$ is inflected form of $word_i$
                $suffix\_list$.append($word_j$.replace($word_i$))
            else
                if $word_j$.startswith($word_i$) and len($word_j$)-len($word_i$) ≥ 4 **then**
                    $diff\_suffix\_list$.append($word_j$.replace($word_i$))
                else
                    continue
    Each $suffix$ in $diff\_suffix\_list$ is manually checked
procedure STEMMING
    for each $word_i$ in $dict$ **do**
        for each $word_j$ in $dict$ **do**
            if $word_j$.startswith($word_i$) **then**
                x $\leftarrow$ $word_j$.replace($word_i$)
                if x in $diff\_suffix\_list$ **then**
                    $word_j$ is removed from $dict$
            else
                continue

n-grams. Initially, we had a list of stopwords for Hindi, but it was in *utf-8* format. Thus, another stopword list was prepared manually in *Romanized* format. This list contains all possible form of a single word, for e.g. '*yun*' and '*yuun*'. The stopwords list contains 307 words in *Romanized* format.

A total of 95,415 unigrams were obtained from the whole corpus after removing the HTML tags and junk characters. The number reduced to 94,960 after removing the stopwords from lyrics. Further, the duplicate characters were removed and this process obtained a total of 75,620 unigrams. Finally, the unsupervised stemming technique was used and it reduced the unigram size to 37,693, though some errors were observed during the stemming process. For example, stemming algorithm trimmed the word '*waaris*' (heir) to '*waar*' (attack) after comparing with the later and after removing the suffix *is*.

4 Methods

4.1 Doc2Vec

Bag-of-words gained immense popularity in the field of text processing, though they have two weaknesses: they lose the ordering of the words and also ignore semantics of the words (Le and Mikolov, 2014). The document level word embeddings have been quite successful in several classification tasks, and it has the advantage over the word embeddings that it is trained to reconstruct linguistic contexts of words. Similarly, Doc2Vec is an extension of word embeddings that learns to correlate labels and words, rather than words with other words.[13] Doc2Vec has been successfully used for several NLP related tasks such as summarization (Pontes et al., 2016), sentiment analysis (Le and Mikolov, 2014) etc.

Initially, we trained all the lyrics using Gensim[14] library. Then, top 10 retrieved vectors for each of the query vectors have been collected for manual checking. We also trained Doc2Vec model on lyrics after stemming all words using the proposed stemming algorithm.

4.2 Self-Organizing Feature Maps

SOFMs are useful for clustering several tasks (Vesanto and Alhoniemi, 2000) and it has been successfully used for information retrieval (Ahuja and Goyal, 2012). SOFMs

[13] https://deeplearning4j.org/doc2vec
[14] https://radimrehurek.com/gensim/models/word2vec.html

are a class of artificial neural networks, which employ competitive learning (Kohonen, 1982). SOFMs cluster similar data without the help of training instances, and hence are said to perform unsupervised learning. The algorithm is started by initializing a set of randomly weighted neurons in the input feature space, and care is taken not to initialize two neurons with identical weights. SOFMs work in two phases, namely self-organizing phase and recall phase. In the self-organizing phase, each neuron's weight vector is matched with an input vector, and the best matching neuron and its neighborhood's weights are adapted to match the selected input. As this kind of learning progresses, input-vectors located far away from each other are mapped to distant neurons. Thus, a grouping of close-by input neurons is formed. In the recall phase, an input vector which is unknown to the SOFMs are matched with all the neurons, and the neighborhood which forms its closest match is associated with that new input vector (Kar et al., 2015).

In self-organizing phase, we considered feature vectors (N-grams) of songs and these were mapped to the neurons to form an SOFMs cluster. In recall phase, a query lyric feature vector was matched with the cluster neighborhoods created during the self-organizing phase. The nearest matching SOFM neighborhood was selected as the set of song ids corresponding to the query song. The detailed steps of SOFMs are given in algorithm 2.

N-gram feature: The n-gram feature plays an important role in information retrieval. The *Term Frequency-Inverse Document Frequency* (*tf-idf*) scores of up to trigrams were considered as feature because the sparsity of the feature vector increases significantly and the results get worse after including higher order n-grams.

4.3 Evaluation

Manual checking is a tedious and time-consuming task requiring human resource. There was no gold standard dataset available for comparing performances of the developed systems. Thus, manual evaluation was performed for calculating similarity between a query and retrieved lyrics. To keep the annotation process simple and reduce the manual checking load, we selected only top ten retrieved lyrics for each query lyric.

We asked the annotator, whether the song is similar to query lyrics or not. Second, whether they would like to listen to retrieved song after listening to the query-song. The annotators were asked to provide a score on a scale from 0 to 1 to each of the ten retrieved lyrics for a single query lyric based on above mentioned points. The retrieved lyric is considered to be matched with the query lyric if the similarity score provided by the annotator is more than 0.7; the threshold was selected experimentally. We wanted a trade off between the system performance and quality of the annotation. This value is selected to reduce the annotation disagreement as well as the subjectivity of the annotators. Two annotators checked each of the results. We also calculated the inter-annotator agreement and it was 87%. Finally, precision (P), recall (R) and F-measure (FM) are calculated based on the manual checking.

Algorithm 2 Pseudo code for SOFMs

1: **procedure** SELF-ORGANIZING PHASE
2: Initialize a neuron field of $k \times k$ dimension, each having $1 \times d$ dimensional weight vector (no two weight vectors would be the same).
3: Select winning weight vector having the least Euclidean distance ($d_{i,j} = \sqrt{\sum_{i=1}^{d} (x_{i,j} - w_{i,j})^2}$) to input vector
4: The winning neuron adapted using $w_{k,j} = w_{k,j} + \eta(x_{k,j} - w_{k,j})$
5: **for** each iteration **do** decrease learning rate η and neighborhood size till convergence
6: **procedure** RECALL PHASE
7: Map input data to nearest clusters centers (weight vectors).

5 Results and Discussion

We have selected a total of 100 query lyrics for testing system performances. For the test, we chose lyrics by searching the top lyrics on the web which are present in the collected dataset. The rest 31,070 lyrics from entire dataset were used for training the system. The systems and their performances with detailed analysis are described below.

5.1 Baseline System

A baseline system was developed for identifying similar lyrics using Fuzzy C-means (FCMs)

clustering algorithm on 34,571 unigrams, and it achieved F-measure of 0.42.

5.2 Doc2Vec based System

We trained Doc2Vec model using all data (i.e. 31,171 lyrics). There were two models, with and without the unsupervised stemming algorithm. We evaluated the Doc2Vec based system using same 100 query lyrics, and the systems achieved F-measures of 0.670 without using the stemming algorithm. Whereas F-measure increased to 0.692 after implementing the stemming algorithm, i.e. an improvement of 0.022 was observed after performing the stemming.

5.3 SOFMs based System

For the SOFMs based system, we changed the distance function from Euclidean to cosine similarity as : $Cosine\ \ Similarity\ (CS) = 1 - \frac{u \cdot v}{||u||^2||v||^2}$

The n-grams were collected after removing stopwords and duplicate characters as well as implementing the unsupervised stemming algorithm. The words having frequency one were also removed from the total 37,693 unigrams and the final unigrams dimension was 34,571. The main reason was that we observed an improvement in the F-measure of 0.008 after removing the unigrams with one frequency. The unigram based system yields F-measure of 0.671 using the Euclidean distance, and there was an improvement of 0.004 when cosine similarity was used for calculating the distance.

After adding bigrams to the above system, the feature dimension increased to 50,321. The bigrams having frequency one is also removed from the lists. The SOFMs based system obtained F-measure of 0.711 using Euclidean distance, and cosine similarity improved the F-measure by 0.007. We developed another system using n-grams up to three, and the feature dimension was 57,321. The similar lyrics retrieval system achieved F-measure of 0.737 using SOFMs with Euclidean distance. An improvement of 0.012 in F-measure was observed using cosine similarity. The detailed results were shown in Table 1.

The higher order n-grams were not included in the study due to computational complexity. The proposed stemming algorithm provides significant computational cost cutting. In fact, we believe that implementing SOFMs for this problem would not be useful if the stemming algorithm was not used. Finally, we removed the repeated lines of *mukhda* from lyrics and developed another system using only unigrams. We observed that the F-measure fell by 0.12 in comparison to system developed using all the words of *mukhda*. Thus, we have not performed any experiments further using this setting.

Algorithms	P	R	FM
FCMs$_U$(Baseline)	0.431	0.410	0.420
Doc2Vec$_{WTS}$	0.670	0.670	0.670
Doc2Vec$_{WS}$	0.691	0.693	0.692
SOFMs$_U$ (EU)	0.721	0.663	0.671
SOFMs$_U$ (CS)	0.721	0.667	0.674
SOFMs$_{UB}$ (EU)	0.727	0.696	0.711
SOFMs$_{UB}$ (CS)	0.732	0.704	0.718
SOFMs$_{UBT}$ (EU)	0.764	0.710	0.737
SOFMs$_{UBT}$ (CS)	0.779	0.718	**0.749**

Table 1: Performances of SOFMs and FCMs based systems.
EU: Euclidean Distance, **CS**: Cosine Similarity, **WTS**: Without Stemming, **WS**: With Stemming, **U**: Unigram, **UB**: Unigram + Bigram, **UBT**: Unigram + Bigram + Trigram

5.4 Discussion

The similar lyrics retrieval systems based on SOFMs and Doc2Vec performed well as compared to baseline system using FCMs. The Doc2Vec based system failed to perform as good as the system developed using SOFMs with n-grams up to three. The Doc2Vec requires a huge amount of training data to train itself and this may be one of the reasons for low performance of Doc2Vec system. Calculating the similarity in the general text is much easier than doing it in the lyrics due to the free word order nature and this may be another reason for poor performance of Doc2Vec based system. It can be stated that the SOFMs work well for clustering similar lyrics. By improving the accuracy of unsupervised stemming algorithm, performances of SOFMs based similar lyrics retrieval system can be increased. Again, adding more number of lyrics can significantly improve the accuracies of such systems.

We investigated the errors in SOFMs based system. We found that there were some mistakes due to spelling variations. For example, the song "*ab to hai tumase har kushii apanii*" does not match with a single song. There are many spelling mistakes in this lyrics such as "*kushii*" (it should

be "*khusii*"), "*apanii*", "*tumase*", "*mashahuur*", "*budanaam*" etc. After removing stopwords, only these words left for the test; thus no match is found with the training dataset. The use of similar words in a different sense makes it harder to identify the similar lyrics in the case of SOFMs based systems. In the case of Doc2Vec, we have not removed stopwords while training the Doc2Vec model; again this model observes the context information rather than only syntactic information (matching the exact word); thus the above lyric has fetched the results.

Searching the similar lyrics was also performed by Mahedero et al. (2005) for Western songs. They used cosine similarity to identify similar lyrics. Another task, identifying similar lyrics based on topics in Japanese songs was performed by Sasaki et al. (2014). They identified topics of lyrics using LDA, and the evaluation was performed using the perplexity. To the best of author's knowledge, no other comparable task has been performed in either in Hindi or Western music.

6 Conclusions and Future Work

We developed a Hindi lyric dataset and implemented several techniques to clean the unstructured data. An unsupervised stemming algorithm was proposed to reduce the number of n-grams. We hope these methods can be used in IR systems for cleaning several unstructured data. The SOFMs based similar lyrics retrieval system achieved the maximum F-measure of 0.749 calculated on 100 query lyrics. We believe that this research will facilitate the development of recommendation in Indian music specifically for Hindi songs.

There are several directions for future work. One of the most immediate tasks is to evaluate performance of the proposed unsupervised stemming algorithm. We used document level word embeddings though, word level embeddings and latent dirichlet allocation (LDA) can be used in future for developing lyrics retrieval systems.

In future, the inter-cluster cosine similarity can be used for automatic evaluation. A weighted score can be assigned to each portions of lyrics (starting, middle, and end) for evaluation.

The mood words from lyrics can be collected using unsupervised approach such as word embeddings and the derived mood information can be used for ranking the results of similar lyrics retrieval system. Ranking the retrieved lyrics is another important factor for recommendation system and is not considered during the evaluation of current system. It is one of the limitations of current developed system and ranking based evaluation can be implemented in future.

Acknowledgments

The work reported in this paper is supported by a grant from the "*Visvesvaraya Ph.D. Scheme for Electronics and IT*" funded by *Media Lab Asia* of *Ministry of Electronics and Information Technology* (*MeitY*), *Government of India*. The authors are also thankful to the anonymous reviewers for their helpful comments.

References

Harika Abburi, Eswar S. A. Akkireddy, Suryakanth Gangashetti, and Radhika Mamidi. 2016. Multimodal sentiment analysis of telugu songs. In *Proceedings of the 4th Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2016)*, pages 48–52.

Sudhir Ahuja and Rinkaj Goyal. 2012. Information retrieval in intelligent systems: Current scenario & issues. *arXiv preprint arXiv:1206.3667*.

Jayson Beaster-Jones and Natalie Sarrazin. 2016. *Music in Contemporary Indian Film: Memory, Voice, Identity*. Taylor & Francis.

James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, et al. 2010. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296. ACM.

Nancy Duncan and Mark Fox. 2005. Computer–aided music distribution: The future of selection, retrieval and transmission. *First Monday*, 10(4).

Yong Han, Li Min, Yu Zou, Zhongyuan Han, Song Li, Leilei Kong, Haoliang Qi, Wenhao Qiao, Shuo Cui, and Hong Deng. 2015. Lrc sousou: A lyrics retrieval system. In *Proceedings of the International Conference of Young Computer Scientists, Engineers and Educators*, pages 464–467. Springer.

Xiao Hu, J Stephen Downie, and Andreas F Ehmann. 2009. Lyric text mining in music mood classification. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 411–416.

Yajie Hu. 2014. *A Model-Based Music Recommendation System for Individual Users and Implicit User Groups*. Ph.D. thesis, University of Miami.

Reshma Kar, Amit Konar, Aruna Chakraborty, Basabdatta Sen Bhattacharya, and Atulya Nagar. 2015. Eeg source localization by memory network analysis of subjects engaged in perceiving emotions from facial expressions. In *International Joint Conference in Neural Networks (IJCNN-2015)*. IEEE.

Peter Knees, Tim Pohle, Markus Schedl, and Gerhard Widmer. 2007. A music search engine built upon audio-based and web-based similarity measures. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 447–454. ACM.

Gopala K. Koduri, Marius Miron, Joan Serrà Julià, and Xavier Serra. 2011. Computational approaches for the understanding of melody in carnatic music. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, pages 263–268.

Teuvo Kohonen. 1982. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Jose P.G. Mahedero, Álvaro MartÍnez, Pedro Cano, Markus Koppenberger, and Fabien Gouyon. 2005. Natural language processing of lyrics. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 475–478. ACM.

Rudolf Mayer, Robert Neumayer, and Andreas Rauber. 2008. Rhyme and style features for musical genre classification by song lyrics. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, pages 337–342.

Cristian Moral, Angélica de Antonio, Ricardo Imbert, and Jaime Ramírez. 2014. A survey of stemming algorithms in information retrieval. *Information Research: An International Electronic Journal*, 19(1).

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2013. Unsupervised approach to hindi music mood classification. In *Proceedings of the Mining Intelligence and Knowledge Exploration*, pages 62–69. Springer International Publishing.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2015. Mood classification of hindi songs based on lyrics. In *Proceedings of the 12th International Conference on Natural Language Processing (ICON- 2015)*, pages 261–267.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016a. Labeling data and developing supervised framework for hindi music mood analysis. *Journal of Intelligent Information Systems*, 48(3):633–651.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016b. Multimodal mood classification - a case study of differences in hindi and western songs. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*, pages 1980–1989.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016c. Multimodal mood classification framework for hindi songs. *Computación y Sistemas*, 20(3):515–526.

Elvys L. Pontes, Juan-Manuel Torres-Moreno, Stéphane Huet, and Andréa C. Linhares. 2016. Tweet contextualization using continuous space vectors: Automatic summarization of cultural documents. In *Proceedings of the CLEF (Working Notes)*.

Shoto Sasaki, Kazuyoshi Yoshii, Tomoyasu Nakano, Masataka Goto, and Shigeo Morishima. 2014. Lyricsradar: A lyrics retrieval system based on latent topics of lyrics. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 585–590.

Markus Schedl, Time Pohle, Peter Knees, and Gerhard Widmer. 2011. Exploring the music similarity space on the web. *ACM Transactions on Information Systems (TOIS)*, 29(3):14:1–14:24.

Rajeswari Sridhar, Manasa Subramanian, B. M. Lavanya, B. Malinidevi, and T. V. Geetha. 2011. Latent dirichlet allocation model for raga identification of carnatic music. *Journal of Computer Science*, 7(11):1711–1716.

Ajay Srinivasamurthy, André Holzapfel, and Xavier Serra. 2014. In search of automatic rhythm analysis methods for turkish and indian art music. *Journal of New Music Research*, 43(1):94–114.

Aniruddha M. Ujlambkar and Vahida Z. Attar. 2012. Mood classification of indian popular music. In *Proceedings of the CUBE International Information Technology Conference*, pages 278–283. ACM.

Juha Vesanto and Esa Alhoniemi. 2000. Clustering of the self-organizing map. *IEEE Transactions on neural networks*, 11(3):586–600.

Chung-Che Wang, Jyh-Shing Roger Jang, and Wennen Wang. 2010. An improved query by singing/humming system using melody and lyrics information. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 45–50.

Menno Van Zaanen and Pieter Kanters. 2010. Automatic mood classification using tf* idf based on lyrics. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 75–80.

Unsupervised Morpheme Segmentation Through Numerical Weighting and Thresholding

Joy Mahapatra*
Technische Universität Darmstadt
Germany
joymahapatra90@gmail.com

Sudip Kumar Naskar
Jadavpur University
Kolkata
sudip.naskar@cse.jdvu.ac.in

Abstract

This paper presents an unsupervised model for morpheme segmentation of words collected from any raw textual corpus of a natural language. The model incorporates a numerical weighting scheme with thresholding technique for finding legitimate morphemes from a given input corpus. Kneedle algorithm is used as a thresholding technique for determining legitimacy of the morphemes. We ran our experiments on five languages – English, Finish, Turkish, German and Bengali, and the model performance is comparable to the state-of-the-art systems.

1 Introduction

Morpheme segmentation of words is an essential part of many linguistic and natural language processing applications. Appropriate morpheme segmentation helps to understand the hidden structure of a language's words and how new words can be built from the existing words. In morpheme segmentation, a word is divided into a stem morpheme and a single affix morpheme (for one-slot morphological languages) or multiple affix morphemes (for multi-slot morphological languages). Stem is also often referred to as base, root, lemma, etc., although they have subtle differences and are used in different contexts. An affix can be of many types; some of the most commonly understood affixes are prefix, suffix, infix, etc. However, for the proposed model, only prefixes and suffixes are considered. Two primary functional types of morphemes exist in morphology: inflectional and derivational morphemes. Inflectional morphemes are affixes that are used to create variant forms of a word in order to signal grammatical information; but they do not change the meaning of the word. Derivational morphemes are affixes that are used to derive new words with new meanings. Both types of morphemes are considered in our work.

The presented model's work principle falls into the category of *Unsupervised Learning of Morphology* (ULM) (Hammarström and Borin, 2011) which usually outputs a morphological structure description of a language from an input raw corpus of that language, provided that the system may need some semi-automatic or manual supervision. The objectives of an ULM based approach can vary. It generally ranges from demand for morphological description of a language to finding lexicon, paradigm list for stems, affix list, same-stem decision, inflectional table and much more. The objective of our proposed model is to discover the stem set and an affix set given a large corpus of a particular language.

Although there are many motivating factors behind ULM from both linguistic and practical point of view (Hammarström and Borin, 2011), the three major motivations are - providing a primary-step for language acquisition, reducing time-consuming manual effort in morphological analysis and language documentation. The first motivation is elicited from the necessity of grabbing primary details and learning basic word structures for a newly observed language. The second motivation is that unsupervised statistical approaches take less amount of time for accomplishing a task without taking much external efforts and resources. The third motivating factor is drawn from a linguistics point of view. It has been observed that in the current world, 80% of the world's languages (almost 7000 total languages) are spoken by only 100,000 speakers or less (Ostler, 2008). It has also been observed that many natural languages are at the verge of extinction (Krauss, 1992). Many linguists fear that with the extinction of such languages, many cultures and valuable information will be lost. They sug-

*Work done while at Jadavpur University.

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 298–304

gest taking help from any immediate quick procedures to restore those almost extinct language details (language documentation). A fast unsupervised approach for morpheme segmentation can provide an essential equipment for language documentation for such languages.

2 Related works

There exist many types of unsupervised morpheme segmentation models and ULM based systems with their own strengths and weaknesses. Hammarström and Borin (2011) classified the ULM models into four underlying types.

The first type emerged as border separation in words through substring frequency determination which explores the idea that if a substring occurs multiple times with other different substrings, then the former substring could be an affix morpheme, whereas the latter ones can be recognized as stem morphemes. After finding such substrings, this type of morphological analysis model tries to define the borders in words. The first-ever ULM based system (Harris, 1955) falls in this category of ULM which is a very popular ULM technique till date. Few researchers (Golcher, 2006; Hammarström, 2009) suggested morpheme segmentation using entropy.

The second type uses grouping and abstracting techniques and they first group all similar morphological words into a particular cluster among many existing ones, then find unique pattern for each cluster of words in such a way that the patterns can reveal all morphemes corresponding to the clusters. his approach is also very common and has multiple implementation examples (Schone, 2001; Yarowsky and Wicentowski, 2000; Wicentowski and Yarowsky, 2002; Wicentowski, 2004; Majumder et al., 2007).

The third ULM based approach (Mayfield and McNamee, 2003; De Pauw and Wagacha, 2007) is quite similar to basic machine learning based approaches. It first represents each word by multiple features and finally stems are separated from the affixes based on the feature values.

The last type of ULM technique is quite similar to the first ULM technique, with a small exception that prior to the border separation, words are categorized based on their phoneme structure (Rodrigues and Cavar, 2007). This ULM technique is applicable for non-concatenative morphology analysis, whereas the rest of the ULM techniques work mainly with concatenative morphological languages. Our proposed approach falls in the first category of ULM.

3 Proposed Method

The proposed morpheme segmentation model takes a raw, unannotated word dataset of an arbitrary language as input. Using a numerical weighting scheme with thresholding strategy, the model ultimately produces a set of stems and a set of affixes. The model also provides the morpheme segmentation of the words. It is to be noted that the model has been proposed and works well with concatenative, one-slot morphological languages (e.g., Bengali), although it is applicable to multi-slot morphological languages (e.g., Turkish, Finnish, etc.).

The proposed morpheme segmentation model for concatenative morphological languages has three basic modules. The first module is responsible for finding all probable initial morphemes (i.e., stems and affixes) from a raw text corpus. The second module scores the morphemes found by the first step. The third module finds out the optimal set of stems and affixes with unsupervised thresholding.

3.1 Morpheme Generation

This module finds out all probable stems and affixes by comparing every word with every other word in a text corpus. For example, by comparing the two words ‘pass’ and ‘passing’, one can easily perceive that ‘pass’ and ‘ing’ could be the stem and affix respectively. For an efficient storing and accessing mechanism of each stem, affix and stem-derived word (i.e., surface word), an implicit matrix ($\mathcal{M}$) type structure is considered, where the matrix columns represent stem-derived words and the rows represent the stems. Each element of the matrix represents an affix (i.e., a prefix or a suffix) or null, which when applied to the corresponding row-word, produces the corresponding column-word. A snapshot of the matrix is shown in Figure 1. Algorithm 1 outlines the process of generating the *stem-affix-word* matrix from the corpus words. To address the scalibility of this alrogithm, we have included a short discussion in Section 4.3.

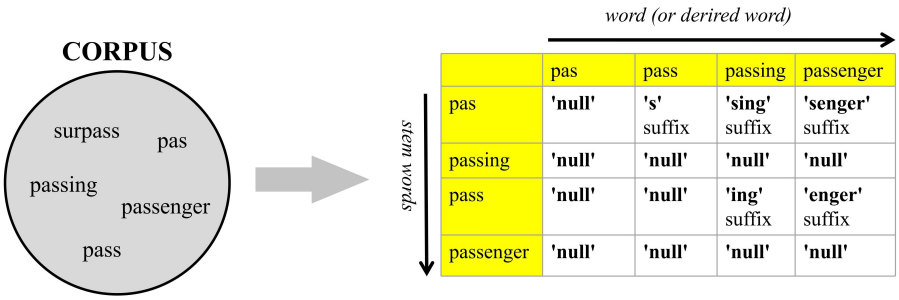


Figure 1: Representation of corpus as stem-affix-word matrix

Input: Raw text corpus $\mathcal{C}$ of language $\mathcal{L}$
Output: $\mathcal{M}$, two-dimentional matrix, with size $|\mathcal{C}| * |\mathcal{C}|$, where $|\mathcal{C}|$ is the number of unique words in $\mathcal{C}$ corpus

begin
  foreach *element* (m) *in matrix* $\mathcal{M}$ **do**
    $m \leftarrow$ '$null$'
  end
  foreach *distinct word* w_1 ***in*** $\mathcal{C}$ **do**
    foreach *distinct word* w_2 ***in*** $\mathcal{C}$ **do**
      if $w_2 = w_1 + a_1$, *where* a_1 *is an affix* **then**
        $\mathcal{M}[w_1][w_2] \leftarrow a_1$
        $\mathcal{M}[w_1][w_2].type \leftarrow$ '$suffix$'
      else if $w_2 = a_1 + w_1$, *where* a_1 *is an affix* **then**
        $\mathcal{M}[w_1][w_2] \leftarrow a_1$
        $\mathcal{M}[w_1][w_2].type \leftarrow$ '$prefix$'
      end
    end
  end
end

Algorithm 1: Generating all possible stems and affixes

3.2 Weighting Morphemes

We propose a weighting scheme that provides a ranking over the morphemes produced by Algorithm 1; the hypothesis is that higher ranked morphemes are likely to be legitimate morphemes of the language. The proposed weighting scheme works in three steps: independent scoring of the affixes, stem scoring through all its possible affixes, and joint stem-affix scoring.

3.2.1 Independent Affix Scoring

In this stage of the weighting scheme, every possible affix found in $\mathcal{M}$ is scored independently. If an affix works as both prefix and suffix, then two different scores are produced for that affix. The independent score for an affix is calculated from the number of different possible stems which appears adjacent to the affix. For an affix (a_x), we refer to this number as its branching factor (bf_{a_x}). Equation 1 shows the calculation of independent score (IS) of a_x from the branching factor of a_x.

$$IS(a_x) = \tanh \beta(bf_{a_x} - 1) \quad (1)$$

This formulation of the affix score (as in Equation 1) was chosen for two major reasons. Firstly, affixes whose branching factor is 1 are canceled out since such affixes carry no or very little significance with regard to the legitimacy of the affix. Secondly, we want high independent score (close to 1) for all affixes above a certain value of branching factor so that affixes with very high branching factors can not dominate over affixes having low branching factors. Although the parameter β needs to be tuned for optimal performance, we chose a value of 2 for β for our experiments. Figure 2 shows the tangent hyperbolic function (cf. equation 1) for varying β values.

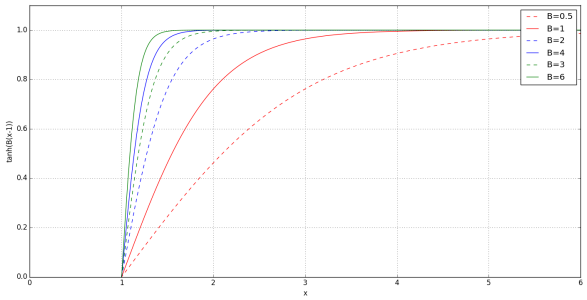


Figure 2: tanh(β(x-1)) function with varying β

Algorithm 1 only considers those stems which appear as words themselves in corpus $\mathcal{C}$. Algorithm 2 alleviates this shortcoming and modifies the matrix $\mathcal{M}$ to discover other possible legitimate stems (that do not appear as words in $\mathcal{C}$) with the help of independent affix scores. Algorithm 2 can also take care of morpheme segmentation in multi-slot morphological languages to some extent.

3.2.2 Affix-Dependent Stem Scoring

This stage determines the affix-dependent score (AdS) for each stem found by Algorithm 2. AdS

Input: Corpus $\mathcal{C}$ and Matrix $\mathcal{M}$ along with the corresponding independent affix scores and types
Output: $\mathcal{M}$, enriched with probable corpus-absent stems and adjusted for multi-slot morpheme segmentation

begin
  /* For multi-slot morphological language */
  foreach a_x *such that* $IS(a_x) == 0$ **do**
    Build a set of sets, S^{a_x}, where, $S^{a_x} = \{\{a_{x_1}, a_{x_2}, ..., a_{x_n}\} : a_x = concat(a_{x_1}.a_{x_2}...a_{x_n})$ and $\forall_i IS(a_{x_i}) \neq 0\}$;
    if $S^{a_x} \neq NULL$ **then**
      Define $mslot\_IS : s_k^{a_x} \in S^{a_x} \rightarrow \mathbb{N}$;
      $mslot\_IS(s_k^{a_x}) = \frac{\sum_{i=1}^{n}\{IS(a_{x_i}) : \text{where } a_{x_i} \in s_k^{a_x}\}}{cardinality(s_k^{a_x})}$;
      $Best_{S^{a_x}} \rightarrow \underset{s \in S^{a_x}}{argmax}\ mslot\_IS(s)$;
      $IS(a_x) \leftarrow mslot\_IS(Best_{S^{a_x}})$;
    end
  end
  /* Generating corpus-absent possible stems */
  foreach *non-zero independent scored affix* a_x *in* $\mathcal{M}$ **do**
    foreach *unique word* w_x *in* $\mathcal{C}$ **do**
      if $w_x = new_{stem} + a_x$ ***or*** $w_x = a_x + new_{stem}$ **then**
        if *no row with* new_{stem} *in* $\mathcal{M}$ **then**
          Make $\mathcal{M}[new_{stem}]$ row;
          $\forall_i(\mathcal{M}[new_{stem}][i] \leftarrow null)$;
        end
        $\mathcal{M}[new_{stem}][w_x] \leftarrow$ 'a_x';
        set $\mathcal{M}[new_{stem}][w_x].type$ accordingly
      end
    end
  end
end

Algorithm 2: Modifying $\mathcal{M}$ for multi-slot morphological languages and corpus-absent stems

is an indicator of the genuineness of a detected stem of being an actual stem. The AdS of a stem depends on its associated affixes in $\mathcal{M}$ and their independent scores. If a stem is associated with more zero independent scored affixes than non-zero independent scored affixes, then the stem loses its genuineness of being a valid stem. The more a stem is associated with non-zero independent scored affixes, the more reliable the stem is.

The AdS of $stem_x$ is computed as in Equation 2 where S is the sum of independent scores of affixes associated to $stem_x$, X and Y represent the number of non-zero and zero independent scored affixes, respectively, associated with $stem_x$, and $\alpha(\geq 1)$ is a penalty factor for associated zero independent scored affixes.

$$AdS(stem_x) = \frac{S}{X + \alpha.Y} \quad (2)$$

Through adjusting the value of α, the affix-dependent score of a stem can be changed with the number of zero independent scored affixes. Large α value highly penalizes this score, whereas low α value do the opposite. For our experiments we fixed α as 2.

3.2.3 Joint Stem-Affix Scoring

$IS(a_x)$ determines the legitimacy of a_x of being an acutal affix. However, the linguistic authenticity of an affix is always estimated along a stem. For example, in English, the '*ing*' suffix holds a high independent score, but the chance of its association with the stem '*k*' (i.e., *k+ing*) is very low compared to the stem '*watch*' (i.e., *watch+ing*), for example. Therefore, a joint scoring mechanism taking into account both affix and stem is required.

The joint stem-affix score ($JSAS$) of $stem_x$ and a_y is computed as in Equation 3.

$$JSAS(stem_x, a_y) = AdS(stem_x) * IS(a_y) \quad (3)$$

3.3 Finding Optimal Set of Stems and Affixes with Unsupervised Thresholding

This is the final operational stage of the proposed model which results in an optimal stem set and an affix set (i.e., paradigm list) from $\mathcal{M}$ based on the $JSAS$ scores. A threshold on $JSAS$ is required for achieving this. For the proposed model, a value of 2 was considered for both β and α. The threshold value ($Threshold_{JSAS}$) for $JSAS$ is determined using the Kneedle algorithm (Satopaa et al., 2011), an unsupervised approach for finding the knee points on curves. The knee points in a tunable system parameter's curve represent advantageous values for that parameter which balance the overall system performance compared to most of the other points in that curve. Unlike other knee points detection approaches, the Kneedle algorithm does not incorporate any system specific information to find out the knee points. This aspect of the Kneedle algorithm helps keep our model almost unsupervised.

3.4 Justification of Our Morpheme Weighting Scheme

Although intuitions behind deriving our morpheme weighting scheme may look like a heuristic procedure, actually, the weighting scheme is

firmly rooted in basic linguistic postulates. We came up with those methods (equations) for the weighting scheme after attending a few conventional linguistic and mathematical rules. The Independent Affix Scoring (IS) method can be justified through the Zipf's empirical law. It has been observed for many years that most of the languages and even random texts follow the Zipf's law (Li, 1992). According to the empirical law, in a large dataset, for every individual word ($word$) the multiplication of its rank in the corpus (r_{word}) and count frequency of the word ($CountFreq_{word}$) remains the same (i.e., $r_{word} * CountFreq_{word} \equiv constant$). Figure 3 shows a sample distribution of the Zipf's law.

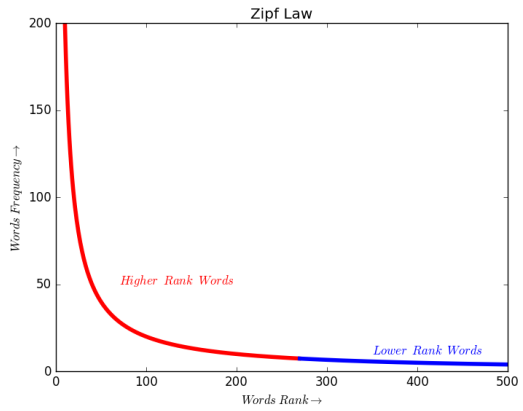


Figure 3: An Ideal Example of Zipf's Law

From this empirical law, it is not difficult to understand that most of the lower ranked words of a large corpus appear for a very few number of times (e,g,. only once or twice). It might so happen that these words with low empirical counts can also possess new affixes. Those affixes will also exist with an unquestionably low count. Therefore, we formulated Equation 1 to select all those low count affixes by assigning them identical weights as the high counted affixes. On the other hand, the second method of the morpheme weighting scheme, Affix-Dependent Stem Scoring (AdS), can be justified when it is seen as a regular mathematical normalization technique with adding denominator penalties for zero independent scored affixes (since zero independent scored affixes are really insignificant). The last method of the weighting scheme, Joint Stem-Affix Scoring ($JSAS$), is nothing but a single objective function comprised of IS and AdS as two distinct objectives.

4 Experiments

4.1 Datasets and Experimental Setup

The proposed method of morpheme segmentation was experimented on five languages – English, Bengali, Finnish, German and Turkish. For English, Turkish, German and Finnish, we used the Morpho-Challenge[1] datasets which provide both raw text corpora as well as gold-standard testsets. The gold-standard datasets mostly contain multi-slot morpheme segmentation samples. The datasets also come with evaluation results of a baseline system (Morfessor) (Creutz and Lagus, 2007). The Morpho-Challenge datasets' training data contains 617,297, 2,338,323, 2,928,030 and 878,036 distinct Turkish, German, Finnish and English words respectively. The test sets contain 1,000 words for each of those four languages. The Dataset also provides a perl script for evaluation on the gold-standard data. For Bengali, we used a gold standard testset (containing 14,034 words) developed in-house and collected a raw corpus (containing 28,927 unique words) by crawling an online Bengali newspaper. Unlike Morpho-Challenge dataset, the Bengali gold-standard data mostly contain single-slot morpheme segmentation examples. The output generated by the system heavily depends on choosing a proper threshold value for $JSAS$ which we determined using the Kneedle algorithm. Figure 4 graphically shows the $JSAS$ score thresholding by Kneedle algorithm for the Bengali dataset.

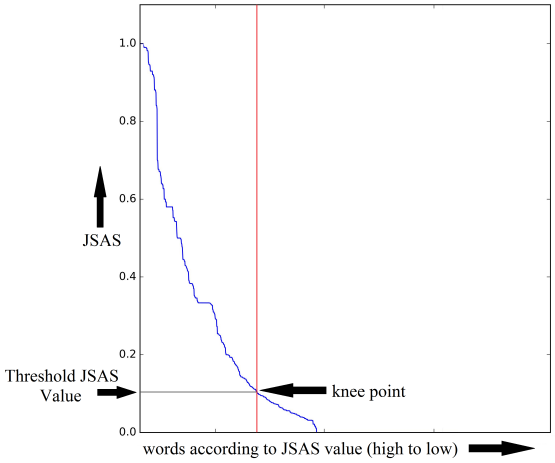


Figure 4: Thresholding using Kneedle algorithm

[1] http://morpho.aalto.fi/events/morphochallenge2010/

Table 1: Evaluation Results

Metric	System	Bengali	English	Turkish	Finnish	German
Precision	B	0.488	0.456	0.421	0.464	0.433
	MB	-	0.813	0.896	0.906	0.828
	P	0.853	0.763	0.695	0.612	0.746
Recall	B	0.842	0.713	0.627	0.675	0.630
	MB	-	0.417	0.177	0.143	0.197
	P	0.724	0.622	0.573	0.542	0.526
F-measure	B	0.617	0.556	0.504	0.550	0.513
	MB	-	0.551	0.296	0.248	0.319
	P	*0.783*	*0.685*	0.628	0.575	*0.617*
	Best	-	0.674	*0.653*	*0.625*	0.508

4.2 Evaluation

System performance was evaluated with precision, recall and f-measure (F1-measure) and the evaluation results are reported in Table 1. We developed a new baseline model which is similar to the proposed model except that it considers $IS_{baseline}(a_x) = bf_{a_x}$ instead of transforming the branching factor through hyperbolic tangent function. Table 1 presents the performance of the newly constructed baseline (*B*), Morfessor baseline (*MB*), the proposed model (*P*) and the best results (*Best*) reported so far on this dataset[2]. The baseline model produces high recall, however, due to absence of a proper thresholding mechanism, it results in low precision and hence low F-measure. We observed that the proposed model shows much better results for single-slot morpheme segmentation compared to multi-slot morpheme segmentation. With the aforementioned set-up, the best performance was observed for Bengali (F-measure 0.783) and the lowest for Finnish (F-measure 0.575). The proposed model outperformed the best results reported so far for English and German on this dataset. Considering that our model is almost unsupervised and it does not require any resources other than a vocabulary, our model results are, overall, comparable with the best results reported on this dataset obtained with semi-supervised approaches.

4.3 Scalabiliy

To keep our morpheme segmentation method scalable towards large vocabulary, we introduced multiple trie data structures to implement the implicit matrix structured shape for storing the stems and affixes. The trie implementation significantly reduces our system running time because of its efficient searching and storing mechanism compared to an ordinary two-dimensional array.

Our model took 1,624.28616 seconds for finding out all possible morpheme segmentations over all the datasets for the mentioned languages. We carried out the entire task on a computer with Intel Core2Duo processor and 4 gigabytes RAM.

[2]http://morpho.aalto.fi/events/morphochallenge2010/results/

5 Conclusions

In this paper we presented an almost unsupervised model for morpheme segmentation given a text corpus. The proposed model uses statistical scoring technique with an unsupervised thresholding algorithm. The model performs bettter on single-slot morpheme segmentation than multi-slot morpheme segmentation. The proposed model yields performance comparable to state-of-the-art performance and outperforms the best results reported so far on English and German.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback. Sudip Kumar Naskar is supported by Media Lab Asia, MeitY, Government of India, under the Young Faculty Research Fellowship of the Visvesvaraya PhD Scheme for Electronics & IT.

References

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1), January.

Guy De Pauw and Peter Waiganjo Wagacha. 2007. Bootstrapping morphological analysis of gikuyu using unsupervised maximum entropy learning. In

Proceedings of the eighth INTERSPEECH conference. Citeseer.

Felix Golcher. 2006. Statistical text segmentation with partial structure analysis. *Proceedings of KONVENS 2006*, pages 44–51.

Harald Hammarström and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350.

Harald Hammarström. 2009. *Unsupervised Learning of Morphology and the Languages of the World*. Ph.D. thesis, University of Gothenburg.

Zellig S Harris. 1955. From phoneme to morpheme. *Language*, 31(2):190–222.

Michael Krauss. 1992. The worlds languages in crisis. *Language*, 68(1):4–10.

Wentian Li. 1992. Random texts exhibit zipf's-law-like word frequency distribution. *IEEE Transactions on information theory*, 38(6):1842–1845.

Prasenjit Majumder, Mandar Mitra, and Dipasree Pal. 2007. Bulgarian, hungarian and czech stemming using yass. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 49–56. Springer.

James Mayfield and Paul McNamee. 2003. Single n-gram stemming. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 415–416. ACM.

Nicholas Ostler. 2008. Is it globalization that endangers languages. UNESCO/UNU Conference: Globalization and Languages: Building our Rich Heritage.

Paul Rodrigues and Damir Cavar. 2007. Learning arabic morphology using statistical constraint-satisfaction models. *AMSTERDAM STUDIES IN THE THEORY AND HISTORY OF LINGUISTIC SCIENCE SERIES 4*, 289:63.

Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In *Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on*, pages 166–171. IEEE.

Patrick John Schone. 2001. *Toward knowledge-free induction of machine-readable dictionaries*. Ph.D. thesis, University of Colorado.

Richard Wicentowski and David Yarowsky. 2002. *Modeling and learning multilingual inflectional morphology in a minimally supervised framework*. Ph.D. thesis, Ph. D. Thesis. Johns Hopkins University, Baltimore, Maryland.

Richard Wicentowski. 2004. Multilingual noise-robust supervised morphological analysis using the wordframe model. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology*, pages 70–77. Association for Computational Linguistics.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216. Association for Computational Linguistics.

Experiments with Domain Dependent Dialogue Act Classification using Open-Domain Dialogue Corpora

Swapnil Hingmire **Apoorv Shrivastava** **Girish K. Palshikar** **Saurabh Srivastava**
{swapnil.hingmire, apoorv.shrivastava}@tcs.com
{gk.palshikar,sriv.saurabh}@tcs.com
TCS Research, Pune, India

Abstract

Dialogue Act (DA) classification plays a major role in the interpretation of an utterance in a dialogue and hence in the development of a dialogue agent. Learning a DA classifier requires large corpora of annotated dialogues which require extensive human efforts and cost. Additionally, nature of dialogue varies based on domain (e.g. tourism, healthcare, finance) as well as the nature of dialogues (e.g. dialogues that involve only queries and responses or dialogues that involve planning or recommendation). Hence, DA classifier trained on a particular corpus may not perform as per the expectations on another domain-*dependent* or task-*dependent* dialogues. In this paper, we propose Conditional Random Field (CRF) based DA classifier, which we train on an open-domain corpus and extend it for a domain-*dependent* corpus by enabling a domain expert to incorporate her domain knowledge in the form of simple rules. Hence, our approach does not need domain-*dependent* labeled corpora. We show the effectiveness of our proposed approach on two real-world datasets.

1 Introduction

Dialogues are an integral part of human interactions, and of arts such as literature, theatre, and films. The presence of discernible common structures in dialogues, despite endless manifestations, has fascinated linguists and artists for ages. With the advent of virtual assistants (chatbots or dialogue agents), there is keen interest in building systems that are capable of having natural and meaningful dialogues with a human user. Dialogues also occur in other major applications, including emails (Cohen et al., 2004), chats (Carpenter and Fujioka, 2011), and web forums (like Wikipedia discussions (Ferschke et al., 2012), students discussion forums (Kim et al., 2010a), comments sections in newspapers, and in community QA systems (Bhatia et al., 2014) like StackOverflow.com.

The theory of dialogue acts provides an important building block in efforts to understand and model structure, function, and flow in dialogues. A *dialogue act* (DA) represents an abstract category of the essential meaning of an utterance (often a sentence or a fragment) in the context of an ongoing dialogue. The *meaning* here usually refers to the agent's intention, the role and relationship of the utterance to the overall dialogue, etc. The *context* of an utterance includes the dialogue state, the mental state, beliefs, and agenda of the human user, and in general, any information contained in previous utterances in the dialogue.

There is a well-accepted set of 43 DAs for English (Stolcke et al., 2000), which have been used to annotate several dialogue corpora; e.g., the human-human telephone English speech Switchboard corpus, Berkeley ICSI Meeting Recorder Digits corpus, etc. Labeling of the DAs to various utterances in a dialogue bring to the fore various relationships among the utterances. For instance, if an utterance is tagged with the DA `YES-NO-QUESTION` then it is likely that the next utterance will have the DA of either `YES-ANSWER`, `NO-ANSWER`, `NON-UNDERSTANDING` or perhaps `OTHER-ANSWER` like *"I don't know"*. These annotated dialogue corpora have been used to build classifiers that automatically identify the DA for any given utterance as part of a given dialogue. DA classification is useful in applications where a computer system is one of the participants in the dialogues; examples: customer help-desk (Bangalore et al., 2006),

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 305–311

tutoring systems (Litman and Silliman, 2004), speech recognition, etc. But it is also used in other applications such as machine translation.

Various machine learning techniques have been used to build classifiers for DA, including HMM (Stolcke et al., 2000), Bayesian Networks (Keizer, 2001), logistic regression (Boyer et al., 2011), language models (Reithinger and Klesen, 1997), multi-layer perceptrons (Wright, 1998), Conditional Random Field (CRF) (Kim et al., 2010b) etc. Recently, several authors have explored deep learning based methods for DA classification (e.g. (Li and Wu, 2016; Khanpour et al., 2016)). An important limitation of these classification approaches is they need a large annotated corpus. More often, they are evaluated on the same domain on which they are trained.

Recently there is increasing trend of building domain specific chat-bots (e.g. domains like Insurance, Finance, Healthcare, IT services helpdesks, Tourism, etc.). It is important to note that conversations in different domains have different characteristics. For example, conversations recorded in IT services help-desks are frequently occurring queries (questions) and their responses, while conversations recorded in Tourism help-desk may involve planning of a tour, purchase of insurance in insurance domain, or recommendation of a product are likely to involve long, and detailed conversations. Additionally, some words have a domain-*dependent* sense, for example, the word "escalation" is used as a synonym to "complaint" in IT services help-desks.

Hence, we hypothesize that a DA classifier trained on an open-domain corpus may not capture characteristics of conversations for different domains and hence, its performance may not be optimal. One way to overcome this problem is to build domain-*dependent* DA classifiers. However, the creation of such classifiers requires huge cost and human efforts. Hence it is important from the practical point of view to build a DA classifier that requires minimum cost and human efforts, at the same time it can be used across multiple domains.

In this paper, we propose a CRF based DA classifier that uses a richer set of features which incorporate lexical, syntactic and semantic information as well as dialogue history. Initially, we learn a DA classifier on an open-domain corpus and then allow a domain expert to incorporate her domain knowledge in the form of simple rules. In our approach, we combine both statistical learning and domain knowledge to build a domain-*dependent* DA classifier.

The paper is organized as follows: In Section 2, we propose our CRF and cue based approach for DA classification. Section 3 discusses evaluation of our proposed DA classifier with respect to a Deep Recurrent Neural Network (RNN) based DA classifier proposed by (Khanpour et al., 2016). In Section 4 we conclude and discuss future prospects of our work.

2 Our Approach

We use cue-based approach for DA classification. Cue phrases are single words, or combinations of words in phrases, that can serve as reliable indicators of some discourse function. A cue-based model uses different sources of knowledge (cues) for detecting a DA such as lexical, collocational, syntactic, prosodic, or conversational-structure cues. This knowledge can then be fed to a machine learning system for training a DA classifier. There is a wide range of features used in DA classification, including the words in each utterance, syntactic information such as Part of Speech (PoS) tags, pragmatic information, including the discourse context as captured by the DAs of preceding utterances, whether there has been a change of speaker, and prosodic information from the acoustic signal if the audio data is available.

Conditional Random Field (CRF) are often applied in machine learning for structured predictions and can be thought of as the sequential version of logistic regression, where logistic regression is a log-linear model for classification, CRF is a log-linear model for sequential labeling. Whereas an ordinary classifier predicts a label for a single sample without regard to neighboring samples, a CRF can take context into account, which is the best match for a problem like conversation analysis as in any conversation most of the utterances are contextually dependent. For example, a lot of information has already been discussed in the conversation till the current utterance, and any new utterance will most likely to keep the already discussed information in mind instead of repeating the information.

We use CRF for training a model with features that provide enough cues for classification of dialogue acts, whether clearly distinguishing DAs like `THANKING` and `APOLOGY`, or closely

related DAs which are hard to distinguish, e.g. all question-related dialogue acts.

2.1 Modeling Steps

The steps for our model creation starts with text cleaning from correction of spelling mistakes and normalization of repeated symbols. In the next step, we change each word to its lemma form, and the corresponding PoS tags are obtained for each of them. In the third step, word bi-grams and PoS bi-grams are also added as features. After adding these features (words, word bi-grams, PoS, PoS bigrams), we introduce a few cue based features for accuracy improvement. We observed that most of the `QUESTION` classes have at least one of the cues for a question, like any one word from WH-Words (what, why, who, where, how) or a question mark "?". Hence, we add a feature to indicate an utterance starting with a WH-Word is likely to be a `QUESTION`. To discriminate `QUESTION` classes further, we add features like presence of WH-Words or collocations based question phrase like *"can I", "are you"*, etc.

We also add separate features for DAs where cues for expressing gratitude, apology or back-channel acknowledgment (like *"Yeah", "okay", "uh-huh"*, etc) are present in an utterance. Additionally, we add a feature for `CONVENTIONAL-OPENING` as the opening utterances of conversations contain words and phrases along with expression of greeting like *"Hello", "Welcome"*, etc. and making it prone to be tagged as `STATEMENT`.

In the end, we created following set of semantic and syntactic features for training of model:

1. Lemmas of words
2. PoS tags
3. PoS tag and word lemma bigrams
4. presence of words that express apology
5. presence of Wh-word
6. presence of words that express gratitude
7. presence of words that indicate start of a conversation
8. presence of a question phrase
9. presence of a question phrase at the beginning of an utterance
10. presence of words that express agreement with the last utterance

3 Experimental Evaluation

We evaluate the performance of our algorithm with Recurrent Neural Network based dialogue act classifier proposed in (Khanpour et al., 2016).

3.1 Datasets

Training datasets:

Since our study focuses on classifying DAs in open-domain conversations, we chose to evaluate our model on Switchboard (SwDA) (Jurafsky et al., 1997) and Dialog State Tracking Challenge 2 (DSTC2[1]) datasets:

- SwDA: The Switchboard corpus (Godfrey et al., 1992) contains 1,155 five-minute, spontaneous, open-domain dialogues. (Jurafsky et al., 1997) revised and collapsed the original DA tags into 43 DAs, which we use to evaluate our model. SwDA has 19 conversations in its test set.

- DSTC2: The Dialog State Tracking Challenge-2 dataset is a conversational dataset of an automated restaurant assistance system and its users, having a total of 2118 different conversations and a total of 19 different user goals which are mapped to 19 different dialogue acts based on similarity of meaning.

Test datasets:

- DSTC2: we used DSTC2 for both training and testing as it is a domain specific dataset.

- Mutual Funds: This dataset contains conversations between customers of an online money management platform and customer service associate through online chat. This dataset is about queries regarding mutual funds transactions through the platform. It contains 26 conversations with total 572 conversational utterances. An example conversation between a customer and a help-desk assistant with manually tagged DAs is given in Table 1

[1] `http://camdial.org/~mh521/dstc/`

Speaker	Utterance	Dialogue Act
Customer	Please assist me in payment of MF	ACTION-DIRECTIVE
Assistant	Hi !	CONVENTIONAL-OPENING
Assistant	This is Jim from ZZZ Mutual Funds Online Assistance.	STATEMENT
Assistant	How may I assist you ?	WH-QUESTION
Customer	I have started mf last month onlly	STATEMENT
Customer	please assist me how can I transfer amount for this month	ACTION-DIRECTIVE
Customer	Hello	CONVENTIONAL-OPENING
Customer	anyone is there ?	STATEMENT
Assistant	Surely I will assist you with the same.	STATEMENT
Assistant	Could you please help me with your registered Email ID and contact number for verification purpose ?	YES-NO-QUESTION
Customer	fname.lname@xyz.com	ABANDONED/UNINTERPRETABLE
Customer	99XX99XX99	ABANDONED/UNINTERPRETABLE
Assistant	Thank you for the details provided.	THANKING
Assistant	Have you schedule any SIP from your mutual fund account	WH-QUESTION
Customer	I dont know much about this	STATEMENT
Assistant	Please provide your PAN No , Date of Birth and Ending 4 Digits of your bank account linked with Myuniverse Investment account	ACTION-DIRECTIVE
Customer	ABCDE0000G	ABANDONED/UNINTERPRETABLE
Customer	DD / MM / YYYY	ABANDONED/UNINTERPRETABLE
Customer	9999	ABANDONED/UNINTERPRETABLE
Assistant	Thank you for the details provided.	THANKING
Assistant	Please be online , I shall check this for you.	STATEMENT
Assistant	Hello sir	CONVENTIONAL-OPENING
Assistant	As checked , you have schedule SIP from your Account	STATEMENT
Customer	Okay	AGREEMENT/ACCEPT
Customer	can you call on my number please	YES-NO-QUESTION
Assistant	Yes sir	YES-ANSWERS
Assistant	Thank you for contacting us.	THANKING
Assistant	Have a nice day.	CONVENTIONAL-CLOSING
Customer	thank you	THANKING

Table 1: An Example Conversation from Mutual Funds Domain

3.2 Experimental Settings

RNN based approach (RNN_{DA})

We used the SwDA and DSTC2 dataset to train RNN_{DA} based model with LSTM layers as described by (Khanpour et al., 2016). All conversations in the training set were preprocessed, and a randomized selection of one-third of them was utilized as a development set to allow the LSTM parameters to be trained over a reasonable number of epochs. We used pre-trained Glove (Pennington et al., 2014) word embeddings of 300 dimension vectors[2]. We used the NN packages provided by (Lei et al., 2015a) and (Lei et al., 2015b). We trained the model with following parameters kept constant (dropout = 0, decayrate = 0.7, dimension of hidden layer = 100, number of layers = 10 and

[2] http://nlp.stanford.edu/data/glove.6B.zip

learning rate = 0.01)

CRF based approach (CRF_{DA})

As both SwDA and DSTC2 datasets are conversational datasets we trained CRF_{DA} model for sequence labeling of dialogue acts. We used CRF implementation from MALLET[3] for training the model.

3.3 Enhancing performance of CRF_{DA}

In the CRF_{DA} classifier for a given sentence output is given as probability distribution across all DAs. We analyzed these output distribution and found that sometimes the correct DA is having slightly less probability than the highest probability DA, so to improve the prediction accuracy we used priority rules for DAs.

Priority Rules:

If the probability difference of top two DAs is within specified threshold and lower probability DA is defined as the high priority then we override the algorithm predicted DA to the high priority DA. For instance, suppose we have defined the threshold as 0.2 probability difference and we have a priority rule defined as: `DA1`→`DA2`, then `DA2` is having higher priority than `DA1` and when in CRF_{DA} output, `DA1` is having higher probability than `DA2` and their probability difference is less than or equal to our threshold 0.2 than the `DA2` (second highest probability dialogue act) is given as prediction in place of `DA1` (highest probability dialogue act).

For example, an utterance with text *"But how come we weren't doing this, say, twenty years ago"* which got tagged with `STATEMENT` and `WH-QUESTION` as top two suggestions with a probability difference of around 0.15 and we can clearly say that the utterance is more of a question than a statement. To handle such cases we defined a priority rule like `STATEMENT`→`WH-QUESTION` with a acceptable probability difference threshold of 0.3 Using this rule whenever a sentence gets `STATEMENT` and `WH-QUESTION` as top two predictions and have a probability difference less than or equal to 0.3 than we change the algorithm prediction from `STATEMENT` to `WH-QUESTION`. We defined few more priority rules based on similar observations.

[3] `http://mallet.cs.umass.edu`

3.4 Analysis of Results

Table 2 show results of our experiments. We can observe the impact of the domain on the performance of both CRF_{DA} and RNN_{DA} classifiers. When we trained RNN_{DA} classifiers using SwDA- an open-domain corpus as a training dataset and evaluated on the domain-*dependent* datasets, the performance was poor. We can also observe that when we trained RNN_{DA} classifiers using DSTC2- a domain-*dependent* corpus and evaluated on the test dataset of DSTC2, the performance is significantly higher when the classifiers are evaluated on SwDA or Mutual Funds dataset. In summary, a RNN_{DA} classifier trained on one corpus of one domain performs poor on dialogues in another domain.

In Table 2, we can observe that CRF_{DA} outperforms RNN_{DA} on both DSTC2 and Mutual Funds dataset when SwDA corpus is used for training. The performance CRF_{DA} is comparable to RNN_{DA} when the dialogues from the same domain are used for both for training and testing. Hence, we can say that performance of both RNN_{DA} and CRF_{DA} is sensitive to the domain of dialogues.

We can also observe in Table 2 that addition of a few manually defined rules to CRF_{DA} classifier (CRF_{DA} + Rules) significantly improves its performance.

4 Conclusions and Future Work

DA classification is an important task in building Dialogue Agents. However, the creation of a sufficiently large tagged dataset for a domain is a highly challenging task as it exerts a high cognitive load on the domain experts (which are likely to be expensive). One approach is to use an open-domain tagged dataset and use it across different domains. In this paper, we proposed a CRF based approach for learning a DA classifier on an open-domain dataset and evaluated it on two different domain-*dependent* datasets. In our approach, we did feature engineering for linguistically motivated features so that the features will capture how *in-general* a dialogue takes place. However, for each domain and further for each domain-specific task, dialogues have different characteristics. To handle such a domain-*dependent* dialogues, we extended our approach through the incorporation of a few easy to define rules which improved the performance of DA classification on

Training Corpus	Test Corpus	RNN_{DA}	CRF_{DA}	CRF_{DA} + Rules
SwDA	SwDA	68.9	66.9	67.1
	DSTC2	21.9	**46.4**	**61.7**
	Mutual Funds	14.5	**58.0**	**63.2**
DSTC2	SwDA	11.1	**21.5**	**21.7**
	DSTC2	94.1	89.8	90.1
	Mutual Funds	33.2	32.9	**43.3**

Table 2: Comparison of DA Classification Accuracy for Different Datasets

domain-*dependent* datasets. In summary, towards the goal of reducing knowledge acquisition overhead in creating domain-*dependent* tagged corpora for different domains, our approach uses existing open-domain corpus to learn a DA classifier and enhances it using a set of manually defined rules.

In future, we would like to do experiments with a few more open-domain and domain-*dependent* dialogues. We would also like to explore transfer learning techniques for DA classification.

References

S. Bangalore, G. Di Fabbrizio, and A. Stent. 2006. Learning the structure of task-driven human-human dialogs. In *Proc. 21st COLING, and 44th ACL*, pages 201–208.

Sumit Bhatia, Prakhar Biyani, and Prasenjit Mitra. 2014. Summarizing online forum discussions – can dialog acts of individual messages help? In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2127–2131, October.

Kristy Boyer, Joseph Grafsgaard, Eun Young Ha, Robert Phillips, and James Lester. 2011. An affect-enriched dialogue act classification model for task-oriented dialogue. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1190–1199, June.

Tamitha Carpenter and Emi Fujioka. 2011. The role and identification of dialog acts in online chat. In *Proc. Workshop on Analyzing Microtext at the 25th AAAI, Conference on Artificial Intelligence*.

W.W. Cohen, V.R. Carvalho, and T.M. Mitchell. 2004. Learning to classify email into "speech acts". In *Proc. Empirical Methods in Natural Language Processing (EMNLP-2004)*, pages 309–316.

Oliver Ferschke, Iryna Gurevych, and Yevgen Chebotar. 2012. Behind the article: Recognizing dialog acts in wikipedia talk pages. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 777–786, April.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, ICASSP'92, pages 517–520, Washington, DC, USA. IEEE Computer Society.

D. Jurafsky, R. Bates, N. Coccaro, R. Martin, M. Meteer, K. Ries, E. Shriberg, A. Stolcke, P. Taylor, and C. Van Ess-Dykema. 1997. Automatic detection of discourse structure for speech recognition and understanding. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 88–95, December.

Simon Keizer. 2001. A bayesian approach to dialogue act classification. In *Proc. BI-DIALOG*.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2012–2021, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Jihie Kim, Jia Li, and Taehwan Kim. 2010a. Towards identifying unresolved discussions in student online forums. In *Proc. NAACL HLT, 2010 Fifth Workshop on Innovative Use of NLP, for Building Educational Applications*, pages 84–91.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2010b. Classifying dialogue acts in one-on-one live chats. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 862–871, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015a. Molding cnns for text: non-linear, non-consecutive convolutions. *arXiv preprint arXiv:1508.04112*.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi S. Jaakkola, Kateryna Tymoshenko, Alessandro Mos-

chitti, and Lluís Màrquez i Villodre. 2015b. Denoising bodies to titles: Retrieving similar questions with recurrent convolutional models. *CoRR*, abs/1512.05726.

Wei Li and Yunfang Wu. 2016. Multi-level gated recurrent neural network for dialog act classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1970–1979, Osaka, Japan, December. The COLING 2016 Organizing Committee.

D. J. Litman and S. Silliman. 2004. ITSPOKE: An intelligent tutoring spoken dialogue system. In *Proc. HLT/NAACL.*

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Norbert Reithinger and Martin Klesen. 1997. Dialog act classification using language models. In *Proc. EuroSpeech-97*, pages 2235–2238.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

Helen Wright. 1998. Automatic utterance type detection using suprasegmental features. In *Proceedings of the International Conference on Spoken Language Processing 1998*, page 1403.

Normalization of Social Media Text using Deep Neural Networks

Ajay Shankar Tiwari
Jadavpur University
India
ajaytiwari0210@hotmail.com

Sudip Kumar Naskar
Jadavpur University
India
sudip.naskar@cse.jdvu.ac.in

Abstract

This paper sets out to investigate ways of normalizing noisy text that appear on social media platforms like Facebook, Twitter, Whatsapp, etc. We proposed a deep learning based approach to text normalization using Recurrent Neural Network (RNN) based Encoder–Decoder architecture with Long Short Term Memory (LSTM). To circumvent the unavailability of suitable large noisy–clean parallel dataset, we developed synthetic datasets. We trained and evaluated the proposed model on our synthetic datasets and the WNUT[1] shared task dataset. The uniqueness of our approach is in the use of synthetic datasets in a transfer learning approach for improving the performance of text normalization based on deep neural models. Our transfer learning based deep neural model produced state-of-the-art results (F1 score 0.9098) outperforming the previous best performing model on the WNUT test set by 7%.

1 Introduction

There is a large quantity of user-generated content on the web, characterized by social media, creativity and individuality, which has created problems at two levels. Firstly, social media text is often unsuitable for various Natural Language Processing (NLP) tasks, such as Information Retrieval, Machine Translation, Opinion Mining, etc., due to the irregularities found in such content. Secondly, non-native speakers of English, older Internet users and non-members of the in-groups often find such texts difficult to comprehend. Prompt use of Internet and the resulting noisy user generated text found in different social media platforms such as social networking sites, blogs, etc., cause a hindrance in understanding casual written English, which often does not conform to the rules of spelling, grammar and punctuation.

In this paper, we present an approach for text normalization of social media text. Our approach uses a sequence to sequence model (Sutskever et al., 2014) in which we tried Recurrent Neural Network (RNN) based encoder-decoder approach (Bahdanau et al., 2014) with Long Short Term Memory (LSTM). The use of LSTMs for text normalization in the present work is motivated by (Sproat and Jaitly, 2016). We take a character based LSTM approach motivated by the work of (Ling et al., 2015) who showed that character based approach is superior to word based approach for neural network based sequence to sequence modelling tasks[2]. Our LSTM model was trained with attention mechanism (Bahdanau et al., 2014).

2 Related Work

Text Normalization is a well known task in the field of NLP, particularly in the Social Media domain. Clark and Araki (2011) provides a detailed survey on the challenges and applications of text normalization in Social Media.

Researchers have shown that text normalization is a major factor in improving performance of NLP intermediate tasks like part-of-speech tagging (Han et al., 2013) and NLP applications like machine translation (Hassan and Menezes, 2013).

Research in text normalization started with spelling correction with noisy channel model (Kernighan et al., 1990; Mays et al., 1991). Since then several different approaches have been proposed by researchers. We report here a few of the most prominent works on text normalization, with a particular focus on social media.

[1] https://noisy-text.github.io/2015/norm-shared-task.html

[2] (Ling et al., 2015) showed improvement on the machine translation task. Text normalization is conceptually very similar to and can be modelled as a machine translation task using noisy–clean parallel corpus..

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 312–321

Statistical Approaches: In the early 1990's, researchers in the AT&T Bell Labs (Kernighan et al., 1990) and IBM Research (Mays et al., 1991) carried out independent work on spelling correction using noisy channel model. This generative model has remained the most dominant and successful approach to text normalization until very recently.

Hassan and Menezes (2013) proposed an approach for normalizing social media text which used random walk framework on a contextual similarity bipartite graph constructed from n-grams sequences, which they interpolated with edit distance . They used the proposed method as a preprocessing step to improve machine translation quality on social media text.

Pennell and Liu (2010) proposed text normalization for text messages (SMS) to make them suitable as input to speech synthesizer. They used statistical classifier which tries to learn when and which character to delete and then reverse the mappings to normalize short text messages.

Sproat et al. (2001) proposed a taxonomy of non-standard words (NSW) and explored several methods like n-gram language models, decision trees, weighted finite state transducers, etc., for text normalization of NSWs. They reported that a systematic class-specific treatment results in improved text normalization.

Choudhury et al. (2007) proposed a Hidden Markov model based text normalization approach for SMS texts and texting language. Aw et al. (2006) used SMT models to normalize noisy SMS text by translating SMS text to Regular English text. Mikheev (2000) solved three major problems in text normalization: sentence boundary disambiguation, disambiguation of capitalized words when they are used in positions where capitalization is expected, and identification of abbreviations.

Deep Learning Based Approaches: Deep learning based approaches have emerged as a competitive alternative in recent years and research have been reported on deep learning based text normalization.

One of the most prominent work on deep learning based approach to text normalization is (Sproat and Jaitly, 2016) which proposed different RNN architectures to normalize texts for text-to-speech (TTS) Systems. The main focus of their work was to normalize *written texts* to their *correct spoken form*. The proposed system used LSTMs and attention based Sequence-To-Sequence models (Bahdanau et al., 2014). Sproat and Jaitly (2017) used the same framework to build their TTS text normalization models for English and Russian and trained their models on huge amount of training data. Their dataset consists of 1.1 billion English words and 290 million Russian words and they reported very high accuracy, over 0.99 for both English and Russian. They also augmented their system with a finite-state transducer (FST) filter to take care of mistakes made by the RNN based model.

Deep Neural Network models suffer from the *Out-Of-Vocabulary* (OOV) problem (Luong et al., 2014), when text normalization is performed using word based approach. Xie et al. (2016) solved this problem by illustrating how character based neural networks are much better in normalizing noisy and user generated texts. They also showed that results can be improved by introducing synthesized errors in a datasets. They showed improvement using noisy text collected from English learner forum.

Leeman-Munk et al. (2015) proposed a model for normalizing noisy text which uses two augmented feed forward networks (Glorot and Bengio, 2010), flagger to identify the word to be normalized and at last a normalizer which provides the correct output for one token at a time.

Chollampatt et al. (2016) showed that neural machine translation models are better in correcting grammatical errors, a task closely related to text normalization, as compared to phrase based statistical machine translation models (Wang et al., 2014).

Other noticeable works in text normalization include the use of adaptive parser-centric strategy (Zhang et al., 2013) to convert noisy texts into grammatically correct texts, unsupervised model using semantic similarity and Re-ranking strategy (Li and Liu, 2014). Torunoğlu and Eryiğit (2014) proposed a cascaded approach for normalizing Turkish text by dividing the main problem into sub problems and solving them one by one. Liu (2012) used character-block level SMT to normalize SMS and Twitter text. Pusateri et al. (2017) reports the use of bi-directional LSTM for the task of *inverse* text normalization, the objective of which is the exact opposite of (Sproat and Jaitly, 2016), i.e. to convert the spoken form token sequence produced by a speech recognizer into written form.

3 System Architecture

3.1 Recurrent Neural Network

The main motive behind using RNN for text normalization is to utilize sequential content. Input to the RNNs is the current information they see as well as the previous information remembered by them at that time. In Fig 1, taken from (Elman, 1991), "BTSXVPE" at the bottom shows the current input and "CONTEXT UNITS" represents the output at the previous step. The decision of recurrent net reached at time step t_i affects the decision it will reach one moment later at time step t_{i+1}. As discussed in (Quast, 2016), RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being dependent on the previous computations. It is often said that recurrent networks have memory. Adding memory to neural networks has a purpose: there is information in the sequence itself and recurrent nets use it to perform tasks. That sequential information is preserved in the recurrent network's hidden state which manages to span many time steps as it cascades forward to affect the processing of each new example. This feature of RNN makes it an efficient approach for normalizing noisy text.

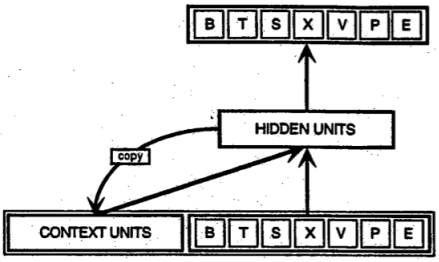


Figure 1: RNN Network Architecture

A drawback of the RNN is that, as the gap between the relevant information and the point where it is needed widens, RNNs fail in remembering the information. This problem with RNN was explored by Bengio et al. (1994). To overcome this problem, LSTM Networks were introduced by Hochreiter and Schmidhuber (1997).

3.2 Long-Short Term Memory Network

As discussed in (Olah, 2015), LSTM networks are another variety of RNNs, mainly used for learning long-term dependencies. In standard RNNs, there is always a chain of repeating modules containing a simple structure.

LSTMs are a fairly simple extension of RNNs. The objective of LSTM can be briefly described into three points as follows.

- **Deciding which information to remember/forget**: The model needs to learn a separate method to forget/remember information when new inputs come in; it needs to know which beliefs to keep and which ones to throw away.
- **Updating new information**: When new input comes in, the model first forgets any long-term information it decides it no longer needs. Then it learns which parts of the new input are worth using and saves them into its long-term memory.
- **Handling long-term dependencies**: Finally, the model needs to learn which parts of its long-term memory are immediately useful and therefore it needs to focus on.

3.3 RNN Encoder–Decoder

RNN Encoder–Decoder models (Cho et al., 2014) can be stated a sequence to sequence mapping between two sequences, learned using two RNNs, one on either side, encoder and decoder, which are trained jointly. For example, in our model, sentences having erroneous words were kept at the encoder side and the corresponding sentences having correctly spelled words were kept at the decoder side. The encoder part receives a sequence and then coverts it into an encoded representation of the sequence, which is further decoded by the decoder to provide the output sequence. RNN encoder-decoder models may contain different cells like GRU or LSTM, or simply RNN. It is quite obvious that the encoded representation of a sequence must be a fixed size vector.

The encoder encodes an input sequence into a fixed-length vector representation and the decoder decodes a given fixed-length vector representation into an output sequence.

3.4 Attention-based Bidirectional RNN Model

Attention Mechanisms in Neural Networks are (very) loosely based on the visual attention mechanism found in humans. Our bidirectional RNN encoder consists of forward and backward RNNs. Using attention mechanism, the decoder gives attention to different parts of the input sequence at each step of the output generation .

4 Data Sets and Resources

As discussed in Section 3.4, we trained a monolingual encoder-decoder based S2S model for our task, where the encoder encodes the input incorrect text and the decoder produces the correct text as output. We consider noisy texts and their corresponding corrected version as a parallel data on which we train our S2S model. However, it is a well known fact that deep neural network models (S2S in our case) typically require large amount of training data. Obtaining such large parallel training data, particularly for the social media domain is a major challenge itself in text normalization research. Therefore, in the absence of such parallel training data, we constructed synthetic data, i.e. artificially developed parallel dataset in which one side consists of sentences having misspelled and noisy words and the other side consists of sentences containing correctly spelled and normalized words. The noisy side (having misspelled words) of the parallel data was created by randomly replacing words in a clean text corpus with the help of four different dictionaries mentioned in Section 4.1. All of these dictionaries consist of parallel list (a hash table) of correct words and their corresponding incorrect noisy versions, where the correct word is stored as a *key* and the values for those keys are one or many misspelled words corresponding to the key.

4.1 Dictionaries

We used Peter Norvig Copus[3], one of the most popular resource in text normalization research, there were containing 7,841 correctly spelled words and their corresponding misspelled version(s). A snapshot of the Peter Norvig Corpus is shown below.

.....
looking: loking, begining, luing, look*2, locking, lucking, louk, looing, lookin, liking
eligible: eligble, elegable, eligable
scold: schold, skold
.....

However, the Peter Norvig corpus is a general domain spelling error corpus, i.e., it is not specifically designed for noisy social media content and as such does not contain spelling error phenomena that are typical to social media text. To include the flavour of errors which occur in social media conversations, we used two spelling error dictionaries provided by the WNUT 2015 shared task[4] on "Normalization of Noisy Text" which contain one to one mapping of incorrect words to correct words. The two dictionaries contain 3,804 and 41,182 [correct, incorrect] word pairs respectively. To make the dataset even noisier, we also constructed another social media spelling error dictionary containing 652 new word pairs by manually observing different Whatsapp Group chat conversations, public comments on Facebook's posts and Facebook Conversations.

4.2 Synthetic Dataset Preparation

We crawled 500K sentences from different news domain websites such as Fox News[5], The Guardian[6], Yahoo News[7] and CNN News[8] and then with the help of Peter Norvig's corpus we prepared a synthetic noisy dataset by replacing some words in each sentence with their corresponding misspelled words, if found in the Peter Norvig corpus. If there exist multiple misspelled versions for the same word in the Peter Norvig corpus, then the choice of misspelled word was taken randomly. Since many of the sentences were very long in the crawled corpus, we broke them (both the original sentence and the corresponding noisy sentence) down to sequences of five-grams in order to keep the sequence length shorter. Here sequence length refers to the number of characters in a sequence. Thus, we created a synthetic parallel dataset, $Synthetic_1$, and Table 1 shows how our parallel synthetic dataset look like, in which the misspelled words are shown as underlined.

We also created another synthetic dataset ($Synthetic_2$) using a Chat Conversation dataset[9] in a similar manner, however, with two noticeable changes. For this dataset we did not split the sequence into n-grams because in this dataset sequence length was not too large as compared to $Synthetic_1$ dataset and we took the help of all the four dictionaries to create the parallel dataset. The chat conversation dataset belongs to the Cornell Movie Dialogue dataset (Danescu-Niculescu-Mizil and Lee, 2011) and it contains conversational data extracted from movie scripts. The dataset was

[3] http://norvig.com/ngrams/spell-errors.txt

[4] https://noisy-text.github.io/2015/norm-shared-task.html

[5] http://www.foxnews.com/

[6] https://www.theguardian.com/international

[7] http://noornotews.yahoo.com/

[8] http://edition.cnn.com/

[9] https://github.com/1228337123/tensorflow-seq2seq-chatbot/tree/master/data

Raw Corpus	
... The government guidance will be reviewed early next year after a period of public comment ...	
Clean Text	**Noisy Text**
... The government guidance will be reviewed government guidance will be reviewed guidance will be reviewed early will be reviwed early next be reviewed early next year reviewed early next year after ...	... The govment guidence will be reviewed guverment guidence we'll be reviewed guidance wil be reviewed erly wiull be reviewed eigly next be reviewed erly enxt yeer reviewed erly nexst year afert ...

Table 1: A snapshot of the $Synthetic_1$ dataset

originally built for building chat systems. We constructed our synthetic data from this raw data. With $Synthetic_2$, our main motive was to build a very noisy dataset containing errors that are typical to social media. Therefore, we replaced as many words as we could find in the two WNUT dictionaries and our social media dictionary. At last, we checked if any of the non-replaced words in a sequence occurs in keys of the Peter Norvig corpus, and if found, they were also replaced with a corresponding (randomly chosen) misspelled word. Thus, we created the parallel dataset $Synthetic_2$ reflecting errors typical of social media text. A snapshot of the $Synthetic_2$ dataset is shown in Table 2.

Other than these synthetic datasets, we also used the standard training dataset released by the WNUT 2015 shared task on "Normalization of Noisy Text". This dataset consists of real Twitter data containing different types of abbreviations used and errors made in social media conversations.

Thus, we ended up with three different datasets[10] – $Synthetic_1$, $Synthetic_2$ and WNUT. Table 3 presents the dataset statistics of all the datasets that we used to train and evaluate our models.

5 Proposed Models

Model 1 (M_1) : This model was trained and tested on the $Synthetic_1$ training and test set, respectively. A batch size of 305 was considered and model was trained for 20 epochs at constant learning rate of 0.001. In this dataset sequence length was 74. Training this model took 20 hours (1 hour/epoch) on a single GPU. This model was built as a general purpose spelling corrector for regular English sentences.

Model 2 (M_2) : From this model onwards, all our models are focused mainly on in social media text. This model was trained and tested on the $Synthetic_2$ training and test set, respectively. The sequence length of was 214 and batch size was kept to 32. The model was trained for 4 epochs as after 4th epoch, loss function started diverging and after making changes in the learning rate, there was no sign of convergence of loss. Learning rate was kept constant for 4 epochs at 0.001. Training this model took 16 hours (4 hour/epoch) on a single GPU.

Model 3 (M_3) : This model was also trained and tested on $Synthetic_2$ dataset, however, the sequence length was kept to 160. We decreased the sequence length so that we could train our model for more number of epochs. Decreasing the sequence length made the training dataset smaller to 1,29,590 sentences, we refer to this dataset as $Synthetic_3$. We were able to train this model for 7 epochs by changing the learning rate form 0.001 to 0.0001 after 4 epochs. Training this model took 17.5 hours (2.5 hour/epoch) on a single GPU.

Model 4 (M_4) : This model was trained and tested on WNUT datasets. Sequence length was kept at 160 and model was trained for 50 epochs and batch size for this experiment was 32. Learning rate was changed from 0.001 to 0.0001 and then back to 0.001, in between the epochs after observing the behaviour of the loss function. Training this model took 2.5 hours (3 minutes/epoch) on a single GPU.

Model 5 (M_5): This model was trained on the merged training datasets of $Synthetic_3$ and

[10]We will release the synthetics datasets and make available for text normalization research upon publication of the paper.

Clean Text	Noisy Text
...	...
Not the hacking and gagging and spitting part.	Not tne hackinq und gaggin nd spittin part.
Please.', "You're asking me out.	Plz.', "You're askin meh out.
That's so cute. What's your name again?"	That's sou cute. What's yur nyam again?"
...	...

Table 2: A snapshot of $Synthetic_2$ dataset

Statistics								
Datasets	Training Set				Test Set			
	Sentences	Words			Sentences	Words		
		Incorrect	Correct	Total		Incorrect	Correct	Total
Synthetic_1	495,000	1,345,500	1,129,500	2,475,000	5,000	15,520	9,480	25,000
Synthetic_2	139,683	953,881	385,792	1,339,673	1,000	7,569	4,101	11,670
Synthetic_3	129,690	856,256	325,456	1,181,781	1,000	6,528	3,594	10,122
WNUT	2,950	19,903	27,482	44,385	1967	11,239	18,182	29,421

Table 3: Dataset Statistics

WNUT. The model was trained for 15 epochs. Loss function was constant at 0.001 and batch size was 256. Training this model took 18.75 hours (1.25 hours/epoch).

Model 6 (M_6): This model was built using transfer learning approach by using the trained weights of model M_3 for further training of the model on the WNUT dataset. Sequence length was kept at 160 and the model was trained for 10 epochs. The batch size was 32. The objective behind using transfer learning approach was to make use of the additional synthetic training dataset and hopefully to improve the system performance on the WNUT test set.

Model 7 (M_7): This is another model built using transfer learning approach, however, in this case training was first carried out on the WNUT datasets for 20 epochs and then the learned weights were further used to retrain the model on the $Synthetic_3$ dataset (10 epochs). Loss function for this model was constant at 0.001 and the batch size was 128. Training this model on the WNUT dataset and the $Synthetic_3$ dataset took 65 minutes (3.25 minutes/epoch) and 6.5 hours (40 minutes/epoch), respectively.

"Negative log Likelihood" was used as the loss function for all the experiments (M_1–M_7) as according to (Lewis and Gale, 1994), this loss function proved to be quite perfect for Sequence to Sequence models. "ADAM" optimizer as described in (Kingma and Ba, 2014) was used for all the experiments. All models mentioned here, used 3 layers on each side encoder & decoder.

6 Experimental Setup

Before loading the datasets for experiments, we padded the data. For faster computation, input sequences were divided into number of batches. Since there were variations in sequence length, we padded the data to make them all having uniform length. "EOS" token was kept at the end of each sequence, to identify its end. Shorter length sequences were padded with trailing zeros. We used word2index dictionary containing *key* as all characters and numbers as their indices. "EOS" and "PAD" were given "0" and "1" index respectively. The length of this dictionary was 98. The synthetic data and the dictionaries mentioned in Section 4.1 will be made publicly available for research upon publication of the paper.

Our models were built using deep learning library *Tensorflow*[11]. TensorFlow allows to efficiently perform specific machine learning number-crunching operations like derivatives on huge matrices . With Tensorflow, processing can be easily distributed across CPU cores, GPU cores, or multiple devices like multiple GPUs and even across a distributed network of computers. *Python*[12] was used for preparation of the dataset as mentioned in section 4.2 and scripting of the models.

7 Results

Since the proposed model can also mistakenly modify some correct words that should not be changed, precision and recall are the most suitable metrics

[11]https://www.tensorflow.org/

[12]https://www.python.org/

for evaluating this scenario (Powers, 2011). Accordingly, we evaluated the proposed models using precision, recall and F1-score. In the context of text normalization, true positives refer to cases where incorrect words are replaced by the correct words, and true negatives represent correct words being left as they are. False positives concern cases when the model replaces a correct word by an incorrect word. False negatives pertain two cases when the model either does not provide any correction or provides a wrong correction for an incorrect word. Table 4 shows the evaluation results of all the experiments mentioned in Section 5. For the benefit of comparison, Table 5 groups these results into two subsets - the ones evaluated on the synthetic test sets and the others evaluated on the WNUT test set. It is to be noted here that the models trained only with the synthetic data (M_1, M_2 and M_3) are not meant to be evaluated with the WNUT test set, however, for the sake of comparison, we also evaluated these models on the the WNUT test set.

Among the experiments carried out only with synthetic datasets, the best result (F1 Score = 0.9205) was achieved with M_1. The other models trained on other synthetic datasets, M_2 and M_3, could not achieve similar results since M_2 and M_3 could not be trained much due to their large sequence length. However, it is to be noted that only the Peter Norvig spelling error corpus was used on news domain data to introduce noise and prepare the $Synthetic_1$ dataset, while all the four dictionaries were employed to introduce noise in conversational data to prepare the $Synthetic_2$ dataset. Therefore, $Synthetic_2$ dataset is much more reflective of social media data and hence more challenging. Among M_2 and M_3, since M_3 was trained on shorter sequences and was also trained for more epochs, it was able to produce better performance than M_2.

Then we evaluated our models on the WNUT dataset, the only standard dataset available for text normalization research. Since the training dataset size was very small, the model (M_4) produced relatively low performance (F1 Score = 0.8223) even after training for 50 epochs. This relatively low performance can be attributed to the very small amount of training data (only 2,950 sentences) in the WNUT dataset; deep learning based models are known to perform poorly than traditional machine learning based methods on small training data. However, this result is only next to the best result (F1 Score = 0.8421) achieved in the WNUT 2015 shared task (Baldwin et al., 2015) in the constrained category.

M_5, trained on a merged training set of $Synthetic_3$ and WNUT, produced better results on both the $Synthetic_3$ as well as WNUT test sets, which can be observed by comparing the performance of M_5 with M_3 and M_4.

Both M_6 and M_7 make use of transfer learning approach. M_6 improves the model performance on the $Synthetic_3$ test set over M_3, however, it could not improve over M_5. On the other hand, M_6 could not beat the performance of M_4 on the WNUT test set. The reason behind this could be that M_6 is essentially a pre-trained model M_3, trained on $Synthetic_3$ and further trained on the WNUT dataset. It is to be noticed however that it provides huge improvement over M_3's performance on the WNUT testset.

We changed the sequence of training in M_7, i.e., first on WNUT and then on $Synthetic_3$ dataset, and also increased the batch size to 128. These changes improved the model performance significantly and provided the best performance on both the $Synthetic_3$ and WNUT test sets. It provided an F1 Score of 0.9098 on the WNUT test set which outperforms the best result (F1 Score = 0.8421) reported in the WNUT shared task (Baldwin et al., 2015).

Table 6 shows a comparison of the results obtained by our systems against the two top performing systems in the WNUT shared task in both constrained and unconstrained track. Surprisingly, unconstrained systems were not able to outperform constrained systems in the WNUT shared task, as is also noted in (Baldwin et al., 2015). However, by making use of our synthetic training data and using a transfer learning approach, we were able to obtain state of the art results on the WNUT dataset.

Our models were able to correctly normalize social media specific errors and abbreviations. For example *"LOL"* was normalized to *"Laughing out Loud"* or *"Lots of Laughs"* depending upon the context, *"GM"* was normalized to *"Good Morning"*, *"k"* to *"ok"* and so on. Among other types of social media errors, *"ohhhhhhhhhhhhhhh"* was normalized to *"oh"*, *"b4"* to *"before"*, *"hiiiiiiii"* to *"hi"*, etc.

Model	Train	Sequence Length	Test	Precision	Recall	F1 Score
M1	$Synthetic_1$	74	$Synthetic_1$	0.9622	0.8822	0.9205
			WNUT	0.1845	0.1756	0.1784
M2	$Synthetic_2$	214	$Synthetic_2$	0.8066	0.7204	0.7610
			WNUT	0.2569	0.2356	0.2457
M3	$Synthetic_3$	160	$Synthetic_3$	0.9102	0.8595	0.8841
			WNUT	0.3096	0.3156	0.3125
M4	WNUT	160	WNUT	0.8558	0.7915	0.8223
M5	Merged datasets ($Synthetic_3$ + WNUT)	160	$Synthetic_3$	0.9366	0.9089	0.9225
			WNUT	0.8569	0.8698	0.8633
M6	Transfer Learning $Synthetic_3 \rightarrow WNUT$	160	$Synthetic_3$	0.9389	0.8856	0.9114
			WNUT	0.8747	0.7260	0.7935
M7	Transfer Learning WNUT $\rightarrow Synthetic_3$	160	$Synthetic_3$	0.9458	0.9056	0.9252
			WNUT	0.9256	0.8945	0.9098

Table 4: Results of different models on synthetic and WNUT datasets

	Synthetic Test Sets		
	Precision	**Recall**	**F1 Score**
M1	0.9622	0.8822	0.9205
M2	0.8066	0.7204	0.7610
M3	0.9102	0.8595	0.8841
M5	0.9366	0.9089	0.9225
M6	0.9389	0.8856	0.9114
M7	0.9458	0.9056	0.9252
	WNUT Test Set		
	Precision	**Recall**	**F1 Score**
M4	0.8558	0.7915	0.8223
M5	0.8569	0.8698	0.8633
M6	0.8747	0.7260	0.7935
M7	0.9256	0.8945	0.9098

Table 5: Results of RNN based LSTM Models on Synthetic and WNUT Test Sets

8 Conclusions & Future work

In this paper we presented a work on text normalization using RNN based encoder-decoder LSTM with an attention mechanism. We illustrate the usefulness of our approach on a variety of noisy datasets - standard real dataset as well as synthetic datasets. We obtained state of the art results on the WNUT dataset using our synthetic training data and a transfer learning approach. Another important contribution of this study is the development of noisy–clean parallel synthetic datasets from user-generated text reflecting both error patterns in regular text as well as social media. The proposed model, with further improvisations, can be useful in the field of social media to make social media communications better and more understandable.

Our next goal is to construct a large real (i.e., not synthetic) social media dataset, similar to WNUT, suitable for training deep learning models which will definitely help to improve text normalization of social media texts. We would also like to explore other deep learning based models for the task.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback. Sudip Kumar Naskar is supported by Media Lab Asia, MeitY, Government of India, under the Young Faculty Research Fellowship of the Visvesvaraya PhD Scheme for Electronics & IT.

References

Aw, A., Zhang, M., Xiao, J., and Su, J. (2006). A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 33–40. Association for Computational Linguistics.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Baldwin, T., De Marneffe, M. C., Han, B., Kim, Y.-B., Ritter, A., and Xu, W. (2015). Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on*

Mode	System	Precison	Recall	F1 Score
Constrained	**NCSU_SAS_NING**	0.9061	0.7865	**0.8421**
	NCSU_SAS_WOOKHEE	0.9136	0.7398	0.8175
	M_4	0.8558	0.7915	0.8223
Unconstrained	**IHS_RD**	0.8469	0.8083	**0.8272**
	USZEGED	0.8606	0.7564	0.8052
	M_5	0.8569	0.8698	*0.8633*
	M_6	0.8747	0.7260	0.7935
	M_7	0.9256	0.8945	*0.9098*

Table 6: Comparison of results between our systems and top two systems of the WNUT shared task

Noisy User-generated Text (WNUT 2015), Beijing, China.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chollampatt, S., Taghipour, K., and Ng, H. T. (2016). Neural network translation models for grammatical error correction. *arXiv preprint arXiv:1606.00189*.

Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S., and Basu, A. (2007). Investigation and modeling of the structure of texting language. *International journal on document analysis and recognition*, 10(3):157–174.

Clark, E. and Araki, K. (2011). Text normalization in social media: progress, problems and applications for a pre-processing system of casual english. *Procedia-Social and Behavioral Sciences*, 27:2–11.

Danescu-Niculescu-Mizil, C. and Lee, L. (2011). Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2Nd Workshop on Cognitive Modeling and Computational Linguistics*, CMCL '11, pages 76–87, Stroudsburg, PA, USA. Association for Computational Linguistics.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3):195–225.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Han, B., Cook, P., and Baldwin, T. (2013). Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):5.

Hassan, H. and Menezes, A. (2013). Social text normalization using contextual graph random walks. In *ACL (1)*, pages 1577–1586.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Kernighan, M. D., Church, K. W., and Gale, W. A. (1990). A spelling correction program based on a noisy channel model. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90, pages 205–210, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Leeman-Munk, S., Lester, J., and Cox, J. (2015). Ncsu_sas_sam: Deep encoding and reconstruction for normalization of noisy text. In *Proceedings of the Workshop on Noisy User-Generated Text at ACL, Beijing, China*, pages 154–61.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc.

Li, C. and Liu, Y. (2014). Improving text normaliza-

tion via unsupervised model and discriminative reranking. In *ACL (Student Research Workshop)*, pages 86–93.

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015). Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Liu, Y. (2012). Improving text normalization using character-blocks based models and system combination.

Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Mays, E., Damerau, F. J., and Mercer, R. L. (1991). Context based spelling correction. *Inf. Process. Manage.*, 27(5):517–522.

Mikheev, A. (2000). Document centered approach to text normalization. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 136–143. ACM.

Olah, C. (2015). Understanding lstm networks.

Pennell, D. L. and Liu, Y. (2010). Normalization of text messages for text-to-speech. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4842–4845. IEEE.

Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

Pusateri, E., Ambati, B. R., Brooks, E., Platek, O., McAllaster, D., and Nagesha, V. (2017). A mostly data-driven approach to inverse text normalization. *Proc. Interspeech 2017*, pages 2784–2788.

Quast, B. (2016). rnn: a recurrent neural network in r. *Working Papers*.

Sproat, R., Black, A. W., Chen, S., Kumar, S., Ostendorf, M., and Richards, C. (2001). Normalization of non-standard words. *Computer speech & language*, 15(3):287–333.

Sproat, R. and Jaitly, N. (2016). Rnn approaches to text normalization: A challenge. *arXiv preprint arXiv:1611.00068*.

Sproat, R. and Jaitly, N. (2017). An rnn model of text normalization. *Proc. Interspeech 2017*, pages 754–758.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Torunoğlu, D. and Eryiğit, G. (2014). A cascaded approach for social media text normalization of turkish. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, pages 62–70.

Wang, Y., Wang, L., Zeng, X., Wong, D. F., Chao, L. S., and Lu, Y. (2014). Factored statistical machine translation for grammatical error correction. In *CoNLL Shared Task*, pages 83–90.

Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., and Ng, A. Y. (2016). Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Zhang, C., Baldwin, T., Ho, H., Kimelfeld, B., and Li, Y. (2013). Adaptive parser-centric text normalization. In *ACL (1)*, pages 1159–1168.

Acronym Expansion: A Domain Independent Approach

Aditya Thakker *
Dwarkadas J. Sanghvi
College of Engineering
aditya.thakker@djsce.edu.in

Suhail Barot *
Dwarkadas J. Sanghvi
College of Engineering
suhail.barot@djsce.edu.in

Sudhir Bagul
Dwarkadas J. Sanghvi
College of Engineering
sudhir.bagul@djsce.ac.in

Abstract

Acronyms are present in usually all documents to express information that is repetitive and well known. But acronyms can be ambiguous because there can be many expansions of the same acronym. In this paper, we propose a general system for acronym expansion that can work on any acronym given some context information it is used in. We present methods for retrieving all the possible expansions of an acronym from Wikipedia and AcronymsFinder.com. We propose to use these expansions to collect the context in which these acronym expansions are used and then score them using a deep learning technique called Doc2Vec. All these things collectively lead to achieving an accuracy of 90.9% in selecting the correct expansion for given acronym on a dataset we scraped from Wikipedia with 707 distinct acronyms and 14,876 disambiguations.

1 Introduction

Acronyms are short descriptors made from important initial letters of a phrase. The phrase here is referred as an expansion of that acronym. Acronyms are used within these documents to shorten complicated or oft-repeated terms.

Acronym usage is becoming more and more common in emails, tweets, blog posts, etc. And with the increasing popularity of mobile devices, the use of acronyms on social platforms has increased even more because typing in these devices is difficult and acronyms provide a succinct way to express information.

Usually, acronyms will be conveniently defined at the point of the first usage, but sometimes a document will omit the definition entirely, assuming the readers familiarity with the acronym. For example, WHO is often used as an acronym for World Health Organization and usually people are expected to know the expansion of it. Or take CSS as an example, most of the documents wont even mention the expansion of CSS because its such a common acronym for Cascading Style Sheets. But CSS can also mean Content-Scrambling System, Closed Source Software, and Cross-Site Scripting.

Also, many natural language processing applications require preprocessing of a document. Text normalization is one of the most important phase of these preprocessing tasks. The basic task of text normalization is to convert non-standard words like numbers, abbreviations, dates, etc. into standard words, though depending on the task and the domain a greater or lesser number of these non-standard words may need to be normalized. In this phase of text normalization, we need to expand all the acronyms in the document. Acronyms are typically ambiguous because several expansions exist for the same acronym as we saw in the example before. For example, Cable News Network and Convolutional Neural Network are both expansions for the common acronym CNN. To disambiguate these acronyms, we can use context paragraphs that surround these acronyms to find the actual expansion. [1]

In our work, we have studied and created an information retrieval system which takes any acronyms along with some context words and then will expand the acronym based on the score it gives to all the possible expansions on the acronym. As shown in the figure, the system will search for all the possible expansions of the given acronym on Wikipedia and Acronymfinder.com. Once it has the list of all the expansions then it will start finding occurrences of those phrases in Wikipedia to get all the contexts in which it is used. Our system will then represent each poss-

[1] * indicates these author is an equal contributor to this work

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 322–327

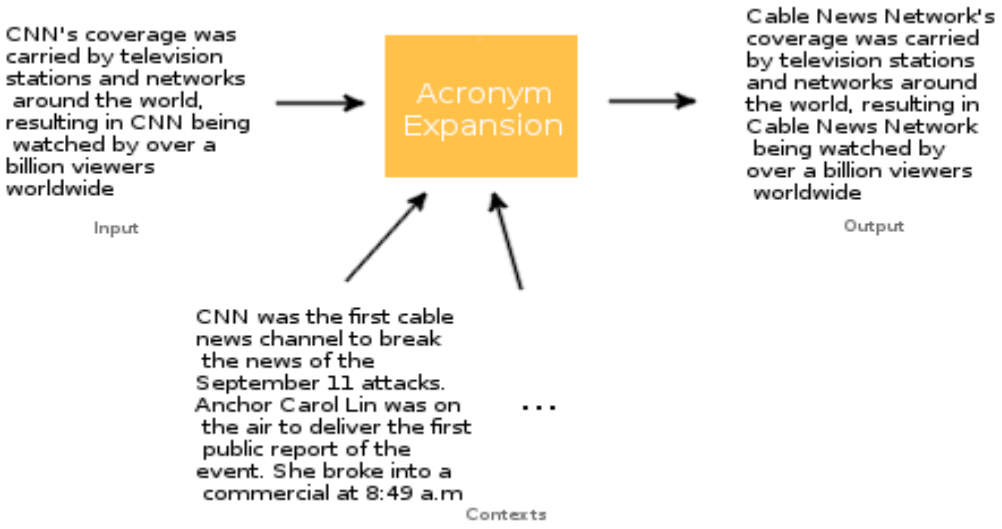


Figure 1: Acronym Exoansion from Input to Output

bile expansion using a deep learning technique called Doc2Vec (Mikolov et al., 2014) in high dimensional vector space. Doc2Vec (Mikolov et al., 2014) which is used in our system can be seen as a distributional semantic representation and this representation is proved to be effective to compute the semantic similarity between words based on the context without any labeled data. The Doc2Vec (Mikolov et al., 2014) embeddings represents the expansions of acronyms in vector space. The placement of each acronym expansion depends on the context that it is used in. Once the system has represented all the possible context vectors associated with each expansion using Doc2Vec (Mikolov et al., 2014), we can pick the expansion whose context vector has the highest cosine similarity score with the input context vector which will then be our expansion for that given acronym.

To the best of our knowledge, we are the first to apply Doc2Vec (Mikolov et al., 2014) embeddings to this task. Experimental results show that our system achieves a comparable accuracy of 90.9% accuracy and is close to humans performance.

Our paper is mainly divided into the following sections:

- In Section 1, we begin with an introduction to the task of acronym expansion and briefly describe our approach.
- In Section 2, we mention the issues with acronym expansion and provide an overview of the past approaches to the same problem.
- In Section 3, we descibe our proposed approach to the task of acronym expansion and the creation of document embeddings from context of acronym usage which is at the core of our model.
- In Section 4, we explain our experimental setup, describe how we gathered the dataset and give results and observations of testing on the datasets.
- In Section 5, we give our conclusions from the experiments and also describe methods to extend our approach to similar problems.

2 Related Work

The task of acronym expansion has been intensively studied by various researchers using supervised learning algorithms. However, the performance of these supervised methods depends on a large amount of labeled data which is extremely difficult to obtain.

In Hippocratic Abbreviation Expansion (Roark et al., 2014) paper, they have used SVM, N-Gram, and many hand-crafted feature engineering techniques to identify the correct expansion of an acronym.

In Acronym-Expansion Recognition and Ranking on the Web (Jain et al., 2007), they use a very similar technique of information retrieval to find all the expansions of any acronyms and then ranked them using co-occurence between acronym and expansion, popularity and reliability of sources.

One other difference between the work we report from much of the recent work cited above is that our work focuses on a more general system to solve the problem. Most of the recent works we have mentioned before are focused on some particular domain and hence use some domain specific techniques to achieve better accuracy. Our system on the other hand only uses the textual data present on Wikipedia to understand the context and outputs the closest expansion similar to input context.

3 Proposed Approach

Owing to the recent success in deep learning frameworks, we sought to apply the techniques to Acronym Expansion problem. But, the main challenge in these approaches is to identify the correct expansion inspite of the many expansions for the same acronym.

We propose to use the vast amount of data available on the internet to identify the correct expansion for any acronym. Our approach involves using Document embeddings to understand the context in which an acronym is used. Document embeddings (Mikolov et al., 2014) are a direct extrapolation of the concept of Word Embeddings (Mikolov et al., 2013). We extract the paragraphs where the acronym was used and supply it to our model. These paragraphs are then embedded in high dimensional vector space, where vector proximity is a direct measure of similarity of context. This concept is explained further in detail in the following sections.

Figure 2: Our Approach

3.1 Crawling Data

As shown in Figure 3, an acronym is given to our system as input. The input is then used to search for all the expansions that we can find for it. To identify, whether any phrase is an expansion of the given acronym, we have made 3 conditions that it must follow:

- The first letters of the words must match the acronym on the sequence
- The words can be separated using space ( ), underscore(_) or dash (-)
- It can consist of stop words in between if the first letters do not match

Implementing these rules, we were able to crawl almost all the expansions that are possible of an acronym.

After finding all the expansions that we could crawl, we had a list of expansions that were possible expansions for the given input. Now, to find the correct expansion, we wanted some contextual data that was used when these expansions were mentioned in any document. Our system would then use the list of expansion to further search for all the occurrences of that expansion and collected some data that surrounds it. This surrounding data is the contextual data that we need to identify the correct expansion of the acronym given to us. The amount of data (words) that we picked surrouding the expansion was of size ranging from 2000-5000 characters (at max). By 2000, we mean that words in 1000 characters before the expansions and words in 1000 characters after the expansion.

It might happen that our system would select an expansion-context pair even if the same expansion has already been fetched. We have purposely allowed it because even if the expansion is same, the context will be different in which the acronym is used. The different contexts for same expansion helps the system to find the correct expansion.

3.2 Model

It has become common practice to use word embeddings (Mikolov et al., 2013) for semantic analysis, the most famous implementations being Googles Word2Vec (Mikolov et al., 2013) and Stanfords GloVe (Pennington et al., 2014). However, researchers have been experimenting, with great success, with sentence/paragraph/document

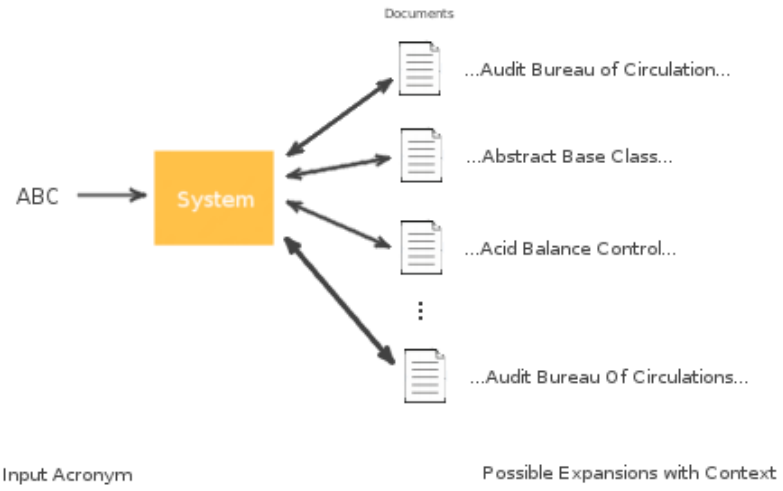


Figure 3: Crawling and finding possible expansions

embeddings - commonly known as thought vectors - for the past few years. Our model is based on Googles Doc2vec (Mikolov et al., 2014) model. It is a neural network architecture that outputs N (number of paragraphs) labelled vectors each of M dimensions.

We have trained our datasets on both the models proposed by Doc2Vec (Mikolov et al., 2014), namely the distributed memory model and distributed bag of words model. The distributed memory model takes into account the context of the surrounding words while predicting a word, while the distributed bag of words model does not.

According to Doc2Vec (Mikolov et al., 2014), given a set of training words, we maximise average log likelihood as:

$$\frac{1}{T}\sum_{t=k}^{T-k}\log p(w_t|w_{t-k},...,w_{t+k})$$

As per their model, prediction is handled by a multiclass classifier (softmax) :

$$p(w_t|w_{t-k},...,w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

We got marginally better results on the Doc2Vec model, as compared to the Distributed Bag of Words model.

As mentioned earlier, our implementation was based on the Doc2vec model.

4 Experiments

We wanted to be absolutely comprehensive in our approach, so we scraped 707 distinct acronyms with their occurences and context in which they had occured.

We got marginally better results on the distributed memory model, as compared to the distributed bag of words model.

For each acronym, we train a model with all the context possibilities. We then calculate the cosine similarity between every input-context and crawled-context pair. Following that, we extract the pair with the highest cosine similarity value. To give some physical intuition, this means that this pair of vectors are the closest together in vector space. We predict that the full form associated with the context selected above is the same as the full form associated with the meaning. Using pythons in built sequence matcher, we match the predicted expansion with the expansion associated with the input context to verify the models prediction and calculate accuracy.

So, for example, if CNN is the acronym at hand, we have one context paragraph and a expansion (Convolutional Neural Network) associated with it, and several crawled context paragraphs (i.e. places on Wikipedia articles where the acronym CNN has occurred). Each context paragraph also has a distinct expansion associated with it. Lets take two distinct context paragraphs, one with a expansion of ”Cable News Network” associated with it, and another with the expansion ”Convolutional Neural Network” associated with it.

We plot all 3 paragraphs in vector space, and calculate the cosine similarity of the input context and all the crawled-contexts pair-wise. So

Doc2Vec Model	Embedding Size.	Context/Source	Length of Source/Context	Traning Epochs	Accuracy
Distributed Bag of Words	500	Context	-	12	88.9%
Distributed Bag of Words	500	Context	-	12	89.7%
Distributed Bag of Words	500	Context	2000	12	90.7%
Distributed Bag of Words	500	Context	2000	12	90.6%
Distributed Bag of Words	200	Source	2000	12	**90.9%**
Distributed Memory	200	Context	5000	12	88.4%
Distributed Memory	750	Context	2000	15	89.7%
Distributed Memory	200	Source	5000	15	86.1%
Distributed Memory	500	Context	5000	15	**90.9%**

Table 1: Results of experiments

here, cos_sim(input_context, crawled_context_1) and cos_sim(input_context, crawled_context_2) are compared. Now we select the pair with the highest cosine similarity, lets say, (input_context, crawled_context). meaning has a full form of "Convolutional Neural Network" associated with it. If crawled_context also has a full form of "Convolutional Neural Network" associated with it, then our model has worked successfully, otherwise not.

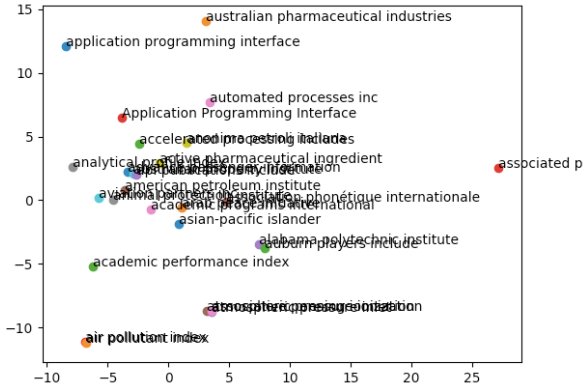


Figure 4: Doc2Vec (Mikolov et al., 2014) Plot for Acronym 'API'

The Figure 4 is an approximate plot of the vector space for the acronym API. This was achieved using Principal Component Analysis. Keeping in mind that 500 dimensions are being condensed to 2 dimensions, this plot is for representation purposes only, and is in no way indicative of the models accuracy.

Using the dataset mentioned before, we ran our model on a total of 14,876 disambiguations for 707 distinct acronyms. We achieved an accuracy of 90.9%.

4.1 Experimental Setup

We use this architecture for the network because of the constraint on the dataset size caused by scarcity of labelled data. We used a NVIDIA 970 GTX GPU and a 4.00 GHz Intel i7-4790 processor with 64GB RAM to train our models. As the datasets in this domain expand, we would like to scale up our approach to bigger architectures. The results obtained on different experiments are given in Table 1. We are able to achieve comparable accuracies without using any domain specific feature engineering.

4.2 Observations

A crawled input for our model ranges from 200 characters to 60,000 characters, as we wanted to simulate real life scenarios as much as possible. A learning rate of 0.025 was found to be ideal, coupled with 12 epochs of training the same model. Less than 10 epochs proved to cause a significant decrease in accuracy due to undertraining. Greater than than 15 epochs of training caused the same problem, but due to overtraining, vectors of 500 dimensions for Distributed Memory model and vectors of 200 dimensions for Distributed Bag of Words model proved to be ideal on our datasets. On smaller paragraphs, smaller dimensions of vectors (100-150) seemed to lead to more accurate predictions, whereas on larger prargraphs, larger dimension vectors(800-1000) worked better.

In some special cases, if an acronym is found in contexts with other acronyms, the models accuracy decreases. For example, in case of acronym "ETC", it can found in context of "European Travel Commission" also. So the cosine similarity score of "European Travel Commission" will be very close to that of "Et Cetera". [2]

5 Conclusion

The experimental results have shown that document embeddings are a promising solution to the acronym disambiguation problem. The results we

[2]Code available at: https://github.com/adityathakker/AcronymExpansion

achieved are stable even without using any handcrafted feature engineering which proves that it's a general data-oriented system.

For further work, we want to try this approach to make recommendation engines that use such contextual data that surrounds any (product) name to identify similar (product) names and recommend them to users.

References

Tomas Mikolov, Quoc V. Le. 2014. Distributed Representations of Sentences and Documents. *Proceedings of the 31 st International Conference on Machine Learning, Beijing, China*,.

Brian Roark, Richard Sproat. 2014. Hippocratic Abbreviation Expansion. *Proceedings of ACL 2014*,.

Alpa Jain and Silviu Cucerzan and Saliha Azzam 2007. Acronym-Expansion Recognition and Ranking on the Web. *IEEE International Conference on Information Reuse and Integration (2007)*,.

Ronan Collobert,Jason Weston,Leon Bottou,Michael Karlen,Koray Kavukcuoglu,Pavel Kuksa. 2011. Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research, 12:2493-2537, 2011*,.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning.. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP:*,.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*,.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *Proceedings of Neural Information Processing Systems*,.

Jeffrey Pennington, Richard Socher, Christopher D. Mannin. 2014. GloVe: Global Vectors for Word Representation. *Empirical Methods in Natural Language Processing (EMNLP)*,.

Sepp Hochreiter , Jrgen Schmidhuber. 1997. Long Short-Term Memory. *Journal Neural Computation archive Volume 9 Issue 8, November 15, 1997* ,.

Diederik Kingma, Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *International Conference for Learning Representations*,.

Ilya Sutskever,James Martens, George Dahl,Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. *Journal of Machine Learning Research*,

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional Recurrent Neural Networks. *IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 45, NO. 11, NOVEMBER 1997*,.

Yoshua Bengio, Patrice Simard, Paolo Frasconi 1994. Learning Long-Term Dependencies with Gradient Descent is difficult. *IEEE Transactions on Neural Networks* ,

Exploring an Efficient Handwritten Manipuri Meetei-Mayek Character Recognition Using Gradient Feature Extractor and Cosine Distance Based Multiclass k-Nearest Neighbor Classifier.

Kishorjit Nongmeikapam
Dept. of CSE
IIIT Manipur, India
kishorjit@iiitmanipur.ac.in

Wahengbam Kanan Kumar
Dept. of ECE
NERIST, Nirjuli, India
wahengbam.kanankumar@gmail.com

Mithlesh Prasad Singh
Dept. of CSE
MIT, Imphal, India
mike77info@gmail.com

Abstract

In this paper, a new approach for efficiently extracting cognition out of a total of 56 different classes of handwritten Manipuri Meetei-Mayek (Indian language) is being described. Although character recognition algorithms has been researched and developed for other Indian scripts, no research work has been reported so far for recognising all the characters of the Manipuri Meetei-Mayek. The work begins with a thorough literature survey of existing works which highlighted the need of a good feature extractor as a pre-requisite for training the classifier. The limitations are experimentally removed using multiple sized cell grids using Histogram of Oriented Gradient (HOG) descriptors as feature extractor. HOG being a gradient based descriptor is very efficient in data discrimination and very stable with illumination variation. For efficient classfication of the HOG features of the Manipuri Meetei-Mayek, the robust k-Nearest Neighbor was tweaked suitably to recognize all the 56 classes of the script. The proposed approach resulted in an overall accuracy of 94.29% with a training time of about 540.81 seconds.

1 Introduction

Handwritten character recognition is increasingly gaining momentum owing to its applicable areas which can significantly reduce time But developing a more dependable approach or more technically 'a system' for recognizing handwritten characters for such regional scripts still poses a challenge to researchers. Moreover, handwritten Meetei-Mayek characters tend to be much more complex in comparison to common English characters due to the presence of modifiers, shape and structure. These factors demand a sophisticated pattern recognition algorithm that will be able to efficiently handle the challenging task of classifying these characters. In this paper, the design of an OCR system for handwritten Manipuri Meetei-Mayek is being discussed. The history and origin of Meetei-Mayek can be found in detail in the literatures by [Wanghemcha ,2007; Mangang, 2003; T.C. Hodson, 1908]. Manipuri or Meeteilon is one of the scheduled language of India and also the official language of Manipur, which is one of the state located in the North-Eastern part of India. The script contains a total of 56 characters which can classified into five different categories: Iyek Ipee/Mapung Iyek which consists of 27 alphabets, Cheitek Iyek (8 symbols), Lonsum Iyek (8 letters), Khudam Iyek (3 symbols) and Cheishing Iyek which consists of 10 numeral figures. The basic characters or the Iyek Ipee only appear as the main character of a word which may be modified by adding one of the extended symbols or Vowel modifiers to produce the required pronunciation. All the original characters of the Manipuri Meetei-Mayek alphabets are drawn, winded and wreathed based on the features of the human anatomy. Accordingly, the names of the alphabets are the names of the different parts of the human body from where they are derived [Mangang, 2003]. The Meetei-Mayek characters for which recognition are performed in the current work is shown in Fig. 1(a) along with the meaning against their names.

2 Related Works

Introduction of Manipuri Meetei Mayek OCR is in the infant stage whereas many research works have already been carried out on other Indian Scripts of different languages. Section 2.1 and 2.2 highlights the research works carried out on popular Indian languages and Manipuri Meetei-Mayek respectively.

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 328–337

2.1 Research Works on other Indian languages

Rani et al. focussed on the problem of recognition related to Gurumukhi script, they used different techniques for extracting features such as projection histogram, background directional distribution (BDD) and zone based diagonal features. These features extraction techniques were classified using SVM classifier as 5-fold cross validation with RBF (radial basis function) kernel. They achieved a very high accuracy of 99.4% using a combination of BDD and diagonal features with SVM classifier. [Rani et al., 2012]. Pal et al. proposed a system for recognizing offline Bangla handwritten compound characters using Modified Quadratic Discriminant Function (MQDF). Using a 5-fold cross validation technique they were able to obtain an accuracy of 85.90% from a dataset of Bangla compound characters containing 20,543 samples [Pal et al., 2007]. Sharma et al. proposed a scheme for unconstrained offline handwritten Devnagri numeral and character recognition using quadratic classifier based on feature obtained from chain code histogram. They were able to achieve an average accuracy of 98.86% for Devanagri numerals and 80.36% for Devanagri characters [Sharma et al., 2008]. Basu et al. presented recognition system for handwritten Bangla alphabet using a 76 element feature set which inluded 24 shadow features, 16 centroid features and 36 longest-run features. The recognition performances achieved for training and test sets were 84.46% and 75.05% respectively [Basu et al., 2005].

2.2 Research Works on Manipuri Meetei-Mayek

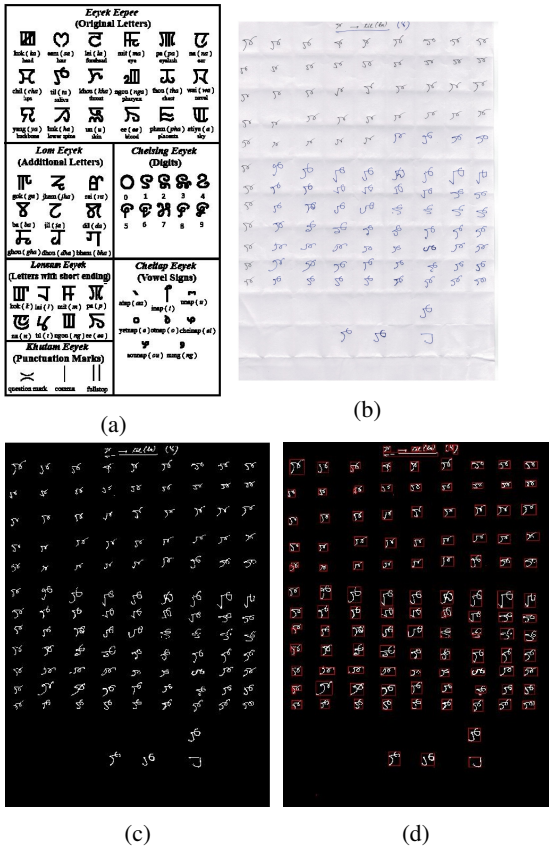
Maring and Dhir described the recognition of Meetei-Mayek numerals for both handwritten as well as printed. Gabor filter was used for feature extraction and classification was carried out using SVM. The experiment was carried out using 14x10 pixel images and overall accuracy of 89.58% and 98.45% were achieved for handwritten and printed respectively [Maring and Dhir, 2014]. Romesh et al. described the design of OCR system for handwritten text in Meitei Mayek alphabets using ANN. The database consists of 1000 samples from which 500 samples were considered as training database and the remaining samples were kept for testing and validation purpose. They observed that success of the system depended on the feature used to represent the character as well as on the segmentation stage of the test image [Romesh et al.,2014] . Chandan and Sanjib in their literature prsented a support vector machine based handwritten numeral recognition system for Manipuri script or Meetei-Mayek. They used various techniques for extracting features such as background directional distribution (BDD), zone-based diagonal, projection histograms and Histogram Oriented features which were then classified using SVM as 5-fold cross validation with RBF kernel. They were able to achieve a maximum accuracy of 95% [Chandan and Sanjib, 2013]. Romesh et al. described a way for simulating and modelling handwritten Meitei Mayek digit using backpropagation neural network approach. They were able to achieve an overall performance of 85% [Romesh et al., 2012]. Thokchom et al. proposed methods for training backpropagation network with probabilistic features, fuzzy features and combination of both features for recognising handwritten Meetei-Mayek characters. They were able to achieve an accuracy of 90.3% for the proposed 27 class classifier neural network with a combination of probabilistic and fuzzy features [Thokchom et al., 2010].

3 System Design

The motivation of this paper is to propose a robust method for classifying offline handwritten Meetei-Mayek characters. The work began with a thorough literature survey of the existing works in Manipuri Meetei-Mayek script. It was realized that so far no literature exist which can successfully or efficiently classify handwritten Meetei-Mayek alphabets and numerals, which is due to the complex nature of the script. However, previous works reported on numerals alone were quite successful as reported in section 2.2 under the heading '*Research Works on Manipuri Meetei-Mayek*'. In the present work, the HOG feature extractor is used prior to the k-NN classification process. A thorough discussion is being highlighted by considering the experimental results for selecting the suitable combination of HOG cell size and the optimal value of neighbor ('k') that can yield the maximum accuracy. By keeping the HOG feature extractor fixed, two different distance metrics that may be used with k-NN classifier is also being compared, which are Euclidean distance met-

ric and Cosine Similarity or distance metric.

To begin with, all the acquired sample images are pre-processed to remove noise as well as for extracting them individually. The pre-processing steps are discussed in section 3.1. As a first approach, based on the work by [Dalal and Triggs, 2005], section 3.2 below describes a procedure for efficiently discriminating feature sets from handwritten Manipuri Meetei-Mayek script using Histogram of Oriented Gradient (HOG) descriptors, the affects of different cell sizes on the length of the extracted features are also studied. Their feature extractor worked by dividing up an image into small spatial regions or cells, each of these regions accumulated a local 1-D edge orientations over pixels of the cell, the combined histogram entries formed the representation. In this work, multiple cell sizes for extracting HOG features have been considered in order to determine which size yielded better results for our current classification problem. The extracted feature vectors were used as training data for the k-NN classifier. Thus, we were able to obtain a significant increase in overall or average accuracy.



(a) (b) (c) (d)

Figure 1: (a) Meetei-Mayek Script (b) A Sample of the Handwritten Character 'TIL (Ta)' (c) Pre-Processed Image (d) Each elements are detected and then encapsulated prior to extraction of each one of them

3.1 Processing the Handwritten Image

In this section, the stages prior to recognition stage is being described.

3.1.1 ***Image Acquisition:*** In this stage raw data is created and collected. A total of 5600 handwritten samples were collected from people having different handwriting styles. Secondly, the image samples were scanned using a scanner and saved as jpeg file. A sample of the acquired handwritten image for the letter 'ꯇ (TIL)' is shown in fig. 1(b).

3.1.2 ***Pre-processing*** In order to make the image suitable for further processing the acquired images must be pre-processed. The term pre-processing refers to removal of any form of noise that is corrupting the useful data so that efficiency as a result of it is not decreased. For a character recognition tasks, a binary image is sufficient to work with, so the input gray image is suitably transformed using thresholding. Morphological erosion is performed so as to close the discontinuities between some letter, square shaped structural element having size equal to 2 is selected for the purpose. Morphological erosion is a simple operators in mathematical morphology which is usually performed in binary images or grayscale images. The purpose of the operation is to erode or decay the boundaries of regions of the foreground pixels (i.e. white pixels), and therefore the areas of foreground pixels shrink in size, and holes within those regions become larger. The morphologically eroded image is finally converted into a binary image[16]. Fig. 1(c) shows the final image after pre-processing.

3.1.3 ***Extracting Individual elements*** Prior to extracting each elements from the binary image so obtained in the previous step, each of them must be labelled so that automatic extraction from them is possible. For this purpose each of the elements are bounded by rectangular boxes. It can be seen from fig. 1(d) that the size of each of the boxes differ due to the fact that some character are bigger than others and vice-versa. The bounding box property for each object is an array having 4 ele-

ment which is formatted as [x, y, w, h], where (x,y) represents the row-column coordinates of the upper left corner of the box. w and h are the width and height of the box. The next step is creating a 4 column matrix that encapsulates all of these bounding box properties together, where each row denotes a single bounding box. It is necessary to define a good illustration of these bounding boxes, and thus a red box is drawn around each characters that was detected. Now, the final task is to extract all of the characters and placing them into a cell array because the character sizes are uneven, so putting this into a cell array will accommodate for the different sizes. A cell array is a type of container used for indexing data called cells, each cells may contain any type of data. Commonly they may contain combinations of text and numbers, or list of strings, or numeric arrays of varying sizes. Now simply looping over every bounding boxes that we have and then extracting the pixels within each of them will result into a character which can be placed in a cell array. Thereafter, using a loop function each of the characters in the cell array are written in to the directory for further usage.

3.2 Feature Extraction using Histogram of Oriented Gradient descriptors

Detecting features in Meetei-Mayek script is a complicated task due the similarity complex of each characters. The very first requirement is a robust feature detector which conforms to the shape or structure of the input image so that characters can be discriminated cleanly. The current study inclines on the issues of feature set extraction from Handwritten Meetei-Mayek Script using the Histogram of Oriented Gradient (HOG) descriptors. The features extracted by multiple cell-sized HOG features are used as training data for multiple classifiers, the details of which are stated in section 3.3.2. The method evaluates normalized histograms of gradient orientation of images in a dense grid. The most simple explanation being because the shapes and appearance of object can be characterized easily by using a distributing the edge detections even without exact knowledge of the corresponding edge positions. It is implemented by dividing up the image window into ”cells” which are small spatial regions. Each cell will accumulate a local 1-D histogram of gradient directions over the cell, and the combined histogram entries form the notation. It is also useful to properly equalize the contrast for improved invariance to shadowing or illumination effects before putting them to use. This feature is achieved by accumulating a measure of ”energy” of the local histogram over somewhat larger spatial ”blocks” or region and then normalizing all of the cell in the block. This is also referred to as *Histogram of Oriented Gradient (HOG)* descriptor. Then, cascading or tiling the detection window with a dense or overlapping grid of HOG descriptors, and using such combined feature vector with a kNN based window classifier will result in a chain detection [Dalal and Triggs, 2005].

Implementation: The implementation of the HOG feature descriptors for Meetei-Mayek script is based on the research work by Dalal and Triggs,2005. The detector has been tested in our Manipuri Meetei-Mayek database which roughly comprises of 56 different classes multiplied by 100 samples each. The training images comprises roughly of 56 different classes times 75 samples each. Pre-processing procedure detailed in section 3.1 is used to segment each of the character samples and finally the images were resized to 50x50 pixels. For testing, the remaining 25 samples for each of the character/class are used to validate how well the classifier performs on data that is different than the training data. Although, this is not the most representative data set, there is enough data to train and test a classifier, and show the feasibility of the approach.

The data which are used for training the classifier are the HOG feature vectors extracted from the input training images. Hence, it is important that the feature vector encodes a sufficient amount of information about the object. With the variation in cell size parameter, the amount of information encoded by each feature vectors can be observed. Each of the pixels in the image calculate a weighted vote for an edge orientation histogram channel. The weighted vote which is based on the orientation of the gradient element are accumulated into bins over local regions which is termed as cells. The orientation bins are specified as a logical scalar and they are evenly spaced from 0 degree - 180 degrees . In this case, the value of scalar less than 0 are placed into a scalar +180 degree value bin. The dark to light versus light to dark transitions contained within some areas of an

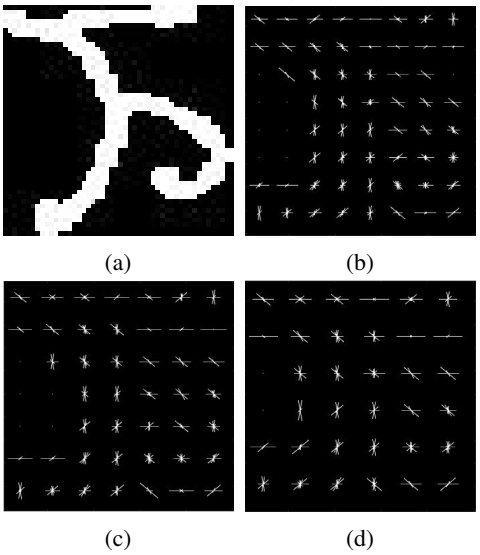


Figure 2: (a) Sample of the Pre-Processed Meetei-Mayek alphabet 'EE-LONSUM' (b) 6x6 HOG cell size (c) 7x7 HOG cell size (d) 8x8 HOG cell size.

image can be differentiated by using signed orientation. The bilinear interpolation of votes between the neighbouring bin centres can reduce aliasing for orientation as well as position. Increasing cell size can be used for capturing large-scale spatial information. It may be noted that cell size is specified as 2-element vectored form in pixels. The suppression of changes in local illumination may be reduced with increasing cell size,i.e. losing minute details as a result of averaging. Therefore, a reduction in the size of blocks will help in capturing the significance of local pixels. However, in actual practice the gradient parameters must be varied by repeatedly training and testing for identifying the optimal parameter settings. For instance, in the current work the optimal block size of HOG feature which must be maintained for efficiently recognizing Meetei-Mayek Characters is explored by considering the cell sizes viz. 6x6,7x7 and 8x8. Fig. 2 shows the features extracted using HOG descriptors for the Meetei-Mayek alphabet '꯵ (EE-LONSUM)'.

The extracted HOG features are returned as 1xN vector. The feature encodes local shape information from regions or from point locations within an image. Where, N is called HOG feature length and is based on the image size and the function parameter values. Let us suppose B_{image} is the number of blocks per image, C is the cell Size, N_b is the number of bins, B_o is the block overlap, B_{size} is the block size, $size_{image}$ is the size of the image. The following equations are used for appropriately deducing the the value of N.

$$N = B_{image}.B_{size}.N_b \tag{1}$$

where,

$$B_{image} = \frac{(\frac{size_{image}}{C} - B_{size})}{B_{size} - B_o} + 1 \tag{2}$$

Table 1 highlights the detected features on Manipuri Meetei-Mayek for different cell sizes. It is important to deduce the dimension of cell size that gives us the best recognition performance when combined with classifiers.

Table 1: Cell size versus HOG Feature length

Cell Size	**Length**
6x6	1764
7x7	1296
8x8	900

3.3 Classification using k-Nearest Neighbor classifier

The k-Nearest Neighbor is an example of a non-parametric type of classifier, it has been used widely as baseline classifying method in many pattern recognition applications. The input to the network consists of k nearest training samples in the feature space, while the output is a membership class. This means that, an n object is duly classified based on a vote of majority among its neighbors, the object is being classified or grouped or assigned the class which is common among its k neighbors nearest to it (it may be noted that k is a small positive integer). In case k equals to 1 then the object is assigned to the nearest neighbor. This technique is also an example of a lazy-learning or instance-based learning in which the functions are considered locally until differed during classification phase. It is also among the simplest of all machine learning tools and yet powerful. It is also quite sensitive to local distribution of data which makes it quite peculiar [Cover and Hart, 1967]

The samples used for training the network are vectors for the multi dimensional space where each of them has a class label. The training phase of the algorithm consists only of storing the HOG features which were extracted from each of the Manipuri Meetei-Mayek samples in section 3.2.

While, in the classification phase, the variable k is user defined, an enlabeled vector is also classified by specifying the class label which is the most frequent and nearest to the query point among the k training samples. The Euclidean distance is the most commonly used distance metric, the optimal value of k depends upon which types of data we are working with. Even though larger values of k has the capability to reduce noise in the classification stage, it can also make the boundaries between the different classes obscure. In multiclass classification problems, it is helpful to choose k to be an odd number as this avoids tied votes. In the current work, for accurately classifying Handwritten Manipuri Meetei-Mayek characters, the value of k are chosen as 1,3,5,7 and 9 .

3.3.1 Distance Metric: Euclidean distance metric is the most popular and widely used similarity measure owing to its simplicity. However, the training images are not all similar necessarily in all features. Due to this limitation, in the current work, the Cosine distance metric is being investigated. A strength of it is that it can normalize all feature vectors to unit length by comparing angle between two vectors. .

Euclidean distance: Euclidean distance computes the ordinary straight line distance between any two points under consideration in the feature space or the Euclidean space. The Euclidean distance between points p and q is the length of the line segment joining them. In the cartesian coordinate system, if $p = (p_1, p_2, ...p_n$ and $q = (q_1, q_2, ...q_n$ are two points in Euclidean n-space, then the distance (d) from p to q, or from q to p is given by the Pythagorean formula:

$$d(p,q) = d(q,p) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \quad (3)$$

Cosine Distance or similarity It is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of $0\deg$ is 1, and it is less than 1 for any other angle. It is thus a judgement of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at $90\deg$ have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where outcome is nearly bounded in [0,1]. The cosine of two non-zero vectors can be derived by using the Euclidean dot product formula:

$$a.b = ||a||_2||b||_2 cos\theta \quad (4)$$

Given two vectors of attribute A and B, the cosine similarity $cos\theta$ is represented using a dot product and magnitude as

$$cos\theta = \frac{A.B}{||A||_2||B||_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (5)$$

where,A_i and B_i are components of vector A and B respectively.

The resulting similarity ranges from -1 meaning exactly opposite, to 1 meaning exactly the same, with 0 indicating orthogonality (decorrelation), and in-between values indicating intermediate similarity or dissimilarity [Manning, 2008].

Table 2: Accuracies for different pairs of HOG sizes with k (neighbors) for the Euclidean metric.

HOG Size	k	Accu. (%)	Training time (s)
6x6	1	88.2	1082.24
6x6	3	92.71	1179.24
6x6	5	89.36	1121.22
6x6	7	91.65	1042.87
6x6	9	90.21	1002.33
7x7	1	91.43	703.34
7x7	3	**94.14**	540.81
7x7	5	91.71	619.62
7x7	7	89.21	767.7
7x7	9	92.07	646.78
8x8	1	92.64	209.86
8x8	3	92.93	371.63
8x8	5	92.36	473.03
8x8	7	91.07	247.33
8x8	9	90.79	206.57

4 Experimental Results and Evaluation

The current section describes the experimental results of the handwritten Manipuri Meetei-Mayek character recognition operation using Multiple HOG feature vector with Multiclass k-NN classifier as described in section 3. The use of Cosine distance Metric based kNN classifier returned a fully trained multiclass, error-correcting output codes (ECOC) model using the training features or HOG descriptors and the class labels in the HOG

Table 3: Accuracies for different pairs of HOG sizes with k (neighbors) for the Cosine metric.

HOG Size	K	Accu. (%)	Training time (s)
6x6	1	93	1011.64
6x6	3	93.93	1141.69
6x6	5	94.07	1108.44
6x6	7	94	1261.14
6x6	9	94.07	967.43
7x7	1	93.29	468.62
7x7	3	**94.29**	502.5
7x7	5	91.14	503.56
7x7	7	93.79	543.37
7x7	9	92.43	527.4
8x8	1	93.07	205.49
8x8	3	93.71	203.03
8x8	5	93.21	231.57
8x8	7	92.71	194.03
8x8	9	92.79	340.98

feature. The *One-versus-one coding scheme* was employed. In this scheme, for each binary learner, one class is positive, another is negative and the software ignores the rest. This design exhaust all combinations of class pair assignments. The number of Binary learners is *K(K-1)/2*, where k is the number of unique class of labels [Escalera et al., 2009], [Escalera et al., 2010]. In the current study, a handwritten character Recognition for Meetei-Mayek Script based on HOG feature descriptors and trained by Cosine distance metric based kNN is successfully implemented. Three different types of HOG Cell Sizes have been considered which were examined for accuracy by training the classifier individually, i.e. 6x6,7x7 and 8x8. For each of the cell size, five different values of k are considered, which are 1,3,5,7 and 9.

In other words, the study pattern is broken up into two areas: firstly, HOG feature descriptors is used with Euclidean distance based kNN, and secondly, HOG feature descriptors is used with Cosine distance based kNN. For each of these two cases, fifteen different combinations each is being used for determining the best combination of k and HOG cell size that yields the best result. Table 2 and 3 shows the different combinations proposed herein. The time taken to train each of the different combinations of classifiers are highlighted in each area. In short, thirty different combinations or classifiers were recorded in our current work. Testing of the 30 different combinations or classifiers were performed and recorded in six different tables, i.e. two times each for 6x6,7x7 and 8x8 HOG cell sizes for Euclidean and Cosine distance metrics. However, owing to the immense size of the tables or the confusion matrices which were recorded for the current work, only the 7x7 HOG cell size with Cosine Distance based kNN success percentage for each of the 56 different classes of the script are shown in table 4 and 5 in the current paper. Some of the characters like '(PA)', '(KHOU)', '(WAI)' in table 4 have very low accuracy in comparison to other characters. For the '(PA)' character the accuracy increased significantly from 68% to 80% which is promising. However, the worst recognition rate is achieved in case of ' 'KHOU' in which the accuracy starts from just 20% and ends at a maximum of 24%. While most of the characters need to be worked on for better efficiency, some others characters like ' ' LAI' ,' 'THOU' and ' ' WAI' also needs an increase in accuracy. Despite the low accuracy readings mentioned above, there are also twenty four (24) cases out of fifty six (56) where the 100% accuracy hold all throughout the different cell sizes viz. - ' (0)', ' (4)', ' (7)', ' (8)', ' (9)', ' (KOK)', ' (TIL)', ' (NGOU)', ' (YANG)', ' (PHAM)', ' (GOK)', ' (RAAI)' , ' (BHAM)', ' (MIT-LONSUM)', ' (PA-LONSUM)', ' (NA-LONSUM)', ' (TIL-LONSUM)', ' (EE-LONSUM)', ' ' (ATAP)', ' (INAP), ' (YET-NAP), ' (OTNAP)', ' (NUNG)', ' (QUESTION MARK)', ' (COMMA)', and ' (FULL-STOP). The overall accuracy achieved by all the 30 different combinations shown in table 2 and 3 highlights a maximum accuracy of 94.29% when k=3 and HOG feature size = 7x7. The time taken to train this particular classifier was 502.5 seconds.

5 Conclusion

In this work, a novel approach for efficiently recognising Handwritten Manipuri Meetei-Mayek Characters is presented by means of comparison between the Euclidean distance Metric based kNN and Cosine distance Metric based kNN for multiple HOG feature descriptors. About 5600 handwritten samples of the 56 different classes of the Manipuri Meetei-Mayek were collected from a group of different people. The samples were then pre-processed to remove the noise in and around the letters followed by extraction of each letters

from the group. The maximum accuracy that we were able to achieve was 94.29% with a training time of just 502.5 seconds by using the Cosine similarity based kNN classification. Thus, we were able to achieve a 0.15% increase in the average recognition rate, along with 38.31 seconds decrease in training time in comparison to the commonly used Euclidean distance based kNN classifier for Manipuri Meetei-Mayek classification.

Therefore, it can be stated that the complex Meeitei-Mayek characters can be efficiently recognised by using the a combination of 7x7 cell-sized HOG descriptors with multiclass Three Nearest Neighbor (3NN) classifier. .

References

Andrew Blais and David Mertz. *An introduction to Neural Networks Pattern Learning with Backpropagation Algorithm*. Gnosis Software, Inc., July 2001.

Anita Rani, Rajneesh Rani and Renu Dhir. *Combination of Different Feature Sets and SVM Classifier for Handwritten Gurumukhi Numeral Recognition*. International Journal of Computer Applications (0975-8887) Vol. 47, No. 18, June 2012.

Cover TM and Hart PE. *Nearest neighbor pattern classification*. IEEE Trans Inf Theory 13(1):2127, 1967.

Chandan Jyoti Kumar and Sanjib Kumar Kalita. Recognition of handwritten Numerals of Manipuri Script. International Journal of Computer Applications (0975-8887), vol. 84, No. 17, Dec. 2013.

Christianini, N., and J. C. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, UK: Cambridge University Press, 2000.

Escalera, S., O. Pujol, and P. Radeva. *Separability of ternary codes for sparse designs of error-correcting output codes*. Pattern Recog. Lett., Vol. 30, Issue 3, 2009, pp. 285297.

Escalera, S., O. Pujol, and P. Radeva. *On the decoding process in ternary error-correcting output codes.* IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 32, Issue 7, 2010, pp. 120134.

Manning CD, Raghavan P, and Schutze H. *An introduction to information retrieval*. Cambridge University Press, Cambridge, 2008.

Maring Kansham Angphun and Renu Dhir. *Recognition of Chesing Iyek/Eeyek-Manipuri Digits using Support Vector Machines*. IJCSIT, vol. 1, Issue 2, April 2014.

N. Dalal and B. Triggs. *Histograms of Oriented Gradients for Human Detection*. In Proc. IEEE Computer Vision and Pattern Recognition, pp. 1-8, 2005.

Ng. Kangjia Mangang. *Revival of a closed account, a brief history of kanglei script and the birth of phoon (zero) in the world of arithmetic and astrology*. Sanamahi Laining Amasung Punshiron Khupham (Salai Punshipham), Lamshang, Imphal, 2003.

N. Sharma, U. Pal,F. Kimura, and S. Pal. *Recognition of Offline handwritten Devnagri characters using quadratic classifier*. in Proc. Indian Conference of Computer Vision Graph. Image Processing, 2006, pp. 808-816.

Romesh Laishram, Pheiroijam Bebison Singh, Thokchom Suka Deba Singh, Sapam Anilkumar, and Angom Umakanta Singh. *A Neural Network Based Handwritten Meetei Mayek Alphabet Optical Character Recognition System*. IEEE International Conference on Computational Intelligence and Computing Research, 2014.

Renato Kresch, and David Malah. *Skeleton-Based Morphological Coding of Binary Images*. IEEE Transaction on image processing, Vol. 7, No. 10, October 1998.

Romesh Laishram, Angom Umakanta Singh, N. Chandrakumar Singh, A. Suresh Singh, H. James. *Simulation and Modelling of Handwritten Meitei Mayek digits using Neural Network Approach*. Proc. of the Intl. Conf. on Advances in Electronics, Electrical, Electrical and Computer Science Engineering - EEC 2012.

Subhadip Basu, Nibaran Das, Ram Sarkar, Mahantapas Kundu, Mita Nasipuri and Dipak Kumar Basu. *Handwritten Bangla alphabet recognition using MLP based classifier*. 2nd National Conference on Computer Processing of Bangla, pp. 285-291, Feb 2005,Dhaka.

Tangkeshwar Thokchom, P.K. Bansal, Renu Vig and Seema Bawa. *Recognition of Handwritten Character of Manipuri Script*. Journal of Computers, Vol. 5, No.10, Oct. 2010.

T.C.Hodson. *The Meitheis*. Low price publications, Delhi, 1908.

U. Pal, T. Wakabayashi and F. Kimura. *Handwritten Bangla Compound Character Recognition using Gradient Feature*. 10th International Conference on Information Technology,2007.

Wangkhemcha Chingtamlen. *A short history of Kangleipak (Manipur) part- II, Kangleipak Historical & Cultural Research Centre*. Sagolband Thangjam Leirak,Imphal, India, 2007.

Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W, Hubbard, L.D. Jackel. *Handwritten digit recognition with a back-propagtion network*. in Advances in Neural Information Processing Systems 2 (NIPS*89), David Touretzky, Ed., Denver, CO, 1990, Morgan Kaufmann.

Table 4: Comparison of accuracy for each classes of the Manipuri Meetei-Mayek among different values of k (where k is the number of neighbors in kNN) for the Cosine based kNN classifier and HOG feature size of 7x7

Sl. No.	Class		Accuracy (%) 7x7 HOG Cell Size				
	Mayek	Notation	k = 1	k = 3	k = 5	k = 7	k = 9
Cheising (Digits)							
1	ꯐ	0	100	100	100	100	100
2	ꯑ	1	88	88	96	92	92
3	ꯒ	2	84	88	84	80	80
4	ꯓ	3	92	100	100	100	96
5	ꯔ	4	100	100	100	100	100
6	ꯕ	5	88	84	84	84	84
7	ꯖ	6	88	92	92	88	84
8	ꯗ	7	100	100	100	100	100
9	ꯘ	8	100	100	100	100	100
10	ꯙ	9	100	100	100	100	100
Eeyek Eepee (Main Alphabet)							
11	ꯀ	KOK	100	100	100	100	100
12	ꯁ	SAM	84	96	96	92	92
13	ꯂ	LAI	80	88	92	92	92
14	ꯃ	MIT	92	92	92	88	88
15	ꯄ	PA	68	80	80	80	80
16	ꯅ	NA	88	88	88	88	92
17	ꯆ	CHEEN	92	92	100	96	100
18	ꯇ	TIL	100	100	100	100	100
19	ꯈ	KHOU	24	24	24	20	20
20	ꯉ	NGOU	100	100	100	100	100
21	ꯊ	THOU	84	84	88	76	68
22	ꯋ	WAI	84	72	68	72	84
23	ꯌ	YANG	100	100	100	100	100
24	ꯍ	HUK	96	96	96	96	96
25	ꯎ	UN	96	96	96	96	100
26	ꯏ	EE	88	88	88	88	88
27	ꯐ	PHAM	100	100	100	100	100
28	ꯑ	ATIYA	100	96	96	96	96

Table 5: Comparison of accuracy for each classes of the Manipuri Meetei-Mayek among different values of k (where k is the number of neighbors in kNN) for the Cosine based kNN classifier and HOG feature size of 7x7 (Contd. from table 4).

Sl. No.	Class		Accuracy (%) 7x7 HOG Cell Size				
	Mayek	Notation	k = 1	k = 3	k = 5	k = 7	k = 9
Lom Eeyek (Addl. Alph.)							
29	ꯘ	GOK	100	100	100	100	100
30	ꯖ	JHAM	96	96	96	96	96
31	ꯔ	RAAI	100	100	100	100	100
32	ꯕ	BAA	100	96	96	96	96
33	ꯖ	JIL	96	96	96	96	96
34	ꯗ	DIL	92	100	92	92	92
35	ꯍ	GHOU	88	92	96	96	96
36	ꯗ	DHOU	84	88	92	92	92
37	ꯙ	BHAM	96	100	100	100	100
Lonsum (Short Alph.)							
38	ꯀ	KOK-LONSUM	96	96	96	92	92
39	ꯂ	LAI-LONSUM	88	92	88	92	92
40	ꯃ	MIT-LONSUM	100	100	100	100	100
41	ꯄ	PA-LONSUM	100	100	100	100	100
42	ꯅ	NA-LONSUM	100	100	100	100	100
43	ꯇ	TIL-LONSUM	100	100	100	100	100
44	ꯀ	NGOU-LONSUM	92	96	96	96	96
45	ꯏ	EE-LONSUM	100	100	100	100	100
Cheising (Vowels)							
46	꯱	ATAP	100	100	100	100	100
47	ꯥ	INAP	100	100	100	100	100
48	ꯨ	UNAP	92	96	96	96	96
49	ꯩ	SOUNAP	96	96	96	96	96
50	ꯣ	YETNAP	100	100	100	100	100
51	ꯤ	OTNAP	100	100	100	100	100
52	ꯦ	CHEINAP	92	92	92	88	92
53	ꯪ	NUNG	100	100	100	100	100
Khutam (Punctuation)							
54	꯫	QUEST. MARK	100	100	100	100	100
56	ꯌ	COMMA	100	100	100	100	100
56	ꯍ	FULLSTOP	100	100	100	100	100

A Modified Cosine-Similarity based log Kernel for Support Vector Machines in the Domain of Text Classification

Rajendra Kumar Roul
Dept of Computer Science
BITS,Pilani-Goa Campus
Zuarinagar
Goa-403726, India
rkroul@goa.bits-pilani.ac.in

Kushagr Arora
Dept of Computer Science
BITS,Pilani- Goa Campus
Goa-403726, India
kushagrarora786@gmail.com

Ishaan Bansal
Dept of EEE
BITS,Pilani- Goa Campus
Zuarinagar
Goa-403726, India
ishaan.bansal.29@gmail.com

Abstract

The popularity of the internet is increasing day-by-day, which makes tough for the end-user to get desired pages from the web in a short time. Text classification, a branch of machine learning can shed light on this problem. State-of-the-art classifier like Support Vector Machines (SVM) has become very popular in the domain of text classification. This paper studies the effect of SVM using different kernel methods and proposes a modified cosine distance based log kernel *(log cosine)* for text classification which is proved as Conditional Positive Definite (CPD). Its classification performance is compared with other CPDs and Positive Definite (PD) kernels. A novel feature selection technique is proposed which improves the effectiveness of the classification and gathers the crux of the terms in the corpus without deteriorating the outcome in the construction process. From the experimental results, it is observed that CPD kernels provide better results for text classification when used with SVMs compared to PD kernels, and the performance of the proposed log-cosine is better than the existing kernel methods.

Keywords: Classification, Conditional positive definite kernels, Cosine distances, Positive definite kernels, Support Vector Machines

1 Introduction

Text classification plays a vital role in the domain of machine learning where the text data is categorized into different groups of similar data items. Many times the present search engine retrieves invalid links and irrelevant web pages for a submitted user query. This weakens the trust of the user on the search engine and thereby degrade its performance. Text classification, a powerful machine learning technique which categorizes an unseen document into its respective predefined class can help in this direction. Two basic classifications of web pages are there: subject-based and genre-based (Qi and Davison, 2009). In subject-based classification, web pages are classified based on their subject or content. Topic hierarchies of web pages are built by this approach. Web pages in genre-based classification are classified into genre or functional related factors, for example, some web pages genres are "multimedia", "home page", "online transaction", and "news headlines". This classification helps users to find their immediate interest from the web without waiting for a long time. There are many classification techniques that exist in real and can be divided into two broad categories: *eager learner* and *lazy learner*. According to eager learner classification technique, the learner built a classification model when the training dataset is given before it receives the test dataset. It can be thought as if the learning model is ready and eager to classify the new test dataset. Examples of this are decision tree, Bayesian network, support vector machine, rule and association based classifier etc. But in lazy learner classification technique, the things are different. Here, instead of building a classification model, it simply stores the training dataset, hence consumes extra space and after seeing the test dataset, it does the classification based on the similarity to the stored training dataset. Examples include k-nearest neighbors (k-NN) and Cased-based reasoning.

Content and *Context* of the web page play major role during the classification process. The sole content of the page including HTML tags, images, text, videos help for classification. Similarly, the hyperlink present in a web page

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 338–347

also decides the page classification. Binary and multi-class are the two basic types of classification exist for classifying the text documents. Binary classification generally categorizes the documents into one of two pre-defined classes whereas multi-class classification handles more than two classes. Classification again can be either single-label or multi-label which is decided depending on the number of labels that is going to be assigned to a document. Exactly one class label is to be assigned to a document in single-label whereas more than one class label is assigned to a document in multi-label classification. For instance, three-class classification means the classification problem consists of three classes say 'Business', 'Sports' and 'Movies'. Many research works has done in the field of web document classification (Aggarwal and Zhai, 2012)(Qiu et al., 2011)(Sebastiani, 2002) (Roul et al., 2017)(Roul and Sahoo, 2017)(Roul et al., 2016)(Roul and Rai, 2016).

Feature selection plays a major role in text classification because selection of important features not only reduces the training time, but also increases the performance of the classifier by reducing the irrelevant features from the corpus. Further, the algorithms used for feature selection are classified into the following three categories:

i. *Filter methods* (Kira and Rendell, 1992) do not use any classifiers for feature selection instead features are selected on the basis of statistical properties. Hence, these methods are fast to compute and capture the usefulness of the feature set which makes them more practical.

ii. *Wrapper methods* (Kohavi and John, 1997) generate different subsets of features based on some algorithms and test each subset using a classifier. To find the score of feature subsets, wrapper methods use a predictive model, whereas filter methods use a proxy measure.

iii. *Embedded methods* (Lal et al., 2006) combines the advantages of both the above two methods and their computational complexity lies between these two methods.

To make the classification process more efficient, a good classifier is required. From the research, it has been observed that the usage of SVM (Cortes and Vapnik, 1995) in text classification has been largely accurate. Many research works on text classification using SVM kernel has been done in the past (Lodhi et al., 2002)(Tong and Koller, 2001)(Zhang et al., 2008)(Maji et al., 2013). Kernel (a similarity function which takes two input feature vectors and find out how similar they are) boost the performance of SVM especially when the number of training documents is more than the number of keywords/features. Kernel can be used on those algorithms which support the inner product that takes the advantage of the nonlinear mapping of features into a high dimensional space with less computational cost (computing the kernel in a higher dimensional space is easy, but computing the feature vector corresponding to the kernel is computationally expensive). Many researchers have worked on SVM kernel (Hamsici and Martinez, 2009)(Hong et al., 2016)(Ponte and Melko, 2017).

Do SVMs work well for text classification?
The theoretical basis for the good performance of SVMs in classifying text documents is suggested by Jaochims (Joachims, 1998) which establishes the following reasons for the same.

i. *High Dimensional Input Space*: By using overfitting protection, SVM does not depend on the number of features and can able to handle a large volume of feature space.

ii. *Few Irrelevant Features*: In text categorization, getting rid of irrelevant features is not of much help, as most features are relevant for classification. So, one cannot easily overcome the problem of high dimensional input space by getting rid of some irrelevant features.

iii. *Sparse Document Vectors*: It has been shown that SVMs are well suited for classification problems with dense concepts and sparse instances.

In this paper, we studied different existing kernels such as RBF, Linear, Polynomial etc. and compare the classification accuracy of SVMs using those kernels techniques. Boughorbel et. al. (Boughorbel et al., 2005) in their work have shown that using SVMs, Conditionally Positive Definite (CPD) kernels provide more accurate

results than Positive Definite (PD) kernels while classifying images. Knowing that SVMs perform well in text classification, here we aim to show that the performance of SVMs using CPD kernels in classifying text document is better than using normal PD kernels. We propose a new kernel based on cosine distances *(log cosine)*, and shown that it is indeed CPD. A novel feature selection technique is proposed in order to reduce the size of the training feature vector which in turn enhances the performance of the classification process. Experimental results on different benchmark datasets show that the usage of log cosine as a kernel in SVM for text classification is better than the existing kernel methods.

The rest of the paper is organized on the following lines: In Section 2, we have discussed the definitions and background details of the proposed approach. Section 3 discusses the proposed approach followed by the experimental analysis discussed in Section 4. Finally, in Section 5, we concluded the work with some future enhancement.

2 Definitions and Basic Preliminaries

This section discusses few important classes of kernels which includes the proposed modified SVM log kernel based on cosine distance and some other background details that are used in the proposed approach.

Definition 1 (Gram Matrix) *Let $K : X \times X \to \mathbb{R}$ be a kernel function. The matrix with entries $K_{ij} := K(x_i, x_j)$ is called Gram matrix or kernel matrix of K with respect to the patterns $x_1, x_2, \cdots, x_n$.*

Definition 2 (Positive Definite Matrix) *A real $n \times n$ matrix with entries K_{ij} is called positive definite matrix if $\sum_{i,j} c_i c_j K_{ij} \geq 0 \; \forall c_i \in \mathbb{R}$.*

Definition 3 (Positive Definite Kernel) *Let X be a non-empty set and K be a symmetric kernel function defined on $X \times X$. If the matrix with elements $K(x_i, x_j)$ is positive definite $\forall n \in \mathbb{N}$ and all $x_1, x_2, \cdots, x_n \in X$ then K is called positive definite kernel.*

Definition 4 (Conditional Positive Definite Kernel) *Let X be a non-empty set. A symmetric kernel K is called conditionally positive definite if $\sum_i^n \sum_j^n c_i c_j K_{ij} \geq 0$ holds $\forall n \in \mathbb{N}$, $x_1, x_2, \cdots, x_n \in X$ and $c_i \in \mathbb{R}$ with $\sum_{i=1}^n c_i = 0$.*

It is needed to show that the negative squared Euclidean distance kernel i.e. $K(x, y) = -\|x - y\|^2$ is conditional positive definite. The proof directly follows from the definition and can be found in (Cowling, 1983). The negative distance kernel is also known as power kernel or triangular kernel. We adapt another conditionally positive definite kernel called *log kernel*, as a part of the study. The log kernel is defined as

$$K_{\log}(x, y) = -\log\left(1 + \|x - y\|^{\beta}\right)$$

On similar lines of the above result, one can easily show that the log kernel is conditionally positive definite.

2.1 Kernel based on Cosine Distances

After suggesting two Conditionally Positive Definite kernels in negative squared distance and log power kernel, we proceed to explore a new kernel method based on cosine distances. The Cosine distance *(CosDis)* is a variant of the Cosine Similarity *(CosSim)* measure and defined as

$$CosDis = 1 - CosSim$$

where $CosSim = \frac{A \cdot B}{|A||B|}$, A and B are two different instances.

We propose a modified kernel function based on cosine distance. There are valid reasons why one may prefer a kernel based on cosine distance rather than the Euclidean distance. The reason for doing so is mainly due to the specific advantages of cosine similarity measures while classifying the text documents (i.e. when the length of two documents are unequal). It is beneficial to abstract out the magnitude of the term vectors so that one can remove the influence of document length. Documents which are clustered using L2-norm instead of direction (i.e. vectors having different directions can be clustered because of their distances from the origin are similar) are highly susceptible to Euclidean distance. When classifying text documents, one uses the angular distance in order to categorize them by their overall sentiments. Relative frequencies of words in the document and across documents are important. Both of these features are exhibited by cosine distance measures. As a result, we propose a modified SVM log kernel based on cosine distance. The

modified kernel based on the cosine distance is defined using equation 1.

$$K(x, y) = -\log(1 + CosDis(x, y)) \quad (1)$$

Before, we prove the above log kernel is conditionally positive definite, we first state the following theorem (Cowling, 1983).

Theorem 1 *If $K : X \times X \rightarrow (-\infty, 0]$ is conditionally positive definite then the following are conditionally positive definite.*

(i) For each α $(0 < \alpha < 1)$, $-(-K)^{\alpha}$.

(ii) $-\log(1 - K)$.

According to Scholkopf (Scholkopf, 2001):

Theorem 2 *If a kernel K is conditionally positive definite then $K+b$ is conditionally positive definite for any constant $b \in \mathbb{R}$.*

Since $K(x, y) = -CosDis(x, y) = -1 + CosSim(x, y)$ and $CosSim(x, y)$ is positive definite, therefore by Theorem 2, K is conditionally positive definite. By applying Theorem 1, we can prove $K(x, y) := -\log(1 + CosDis(x, y))$ is conditionally positive definite. So our proposed cosine based log kernel is conditionally positive definite.

2.2 Term Frequency and Inverse Document Frequency

Term Frequency (TF) measures how often a term t occurs in a document d whereas Inverse Document Frequency (IDF) measures the importance of t in the entire corpus P. TF-IDF (Sparck Jones, 1972) is a technique which finds the importance of terms in a document based on how they appear in the corpus. The TF-IDF is calculated using equation 2.

$$TF\text{-}IDF_{t,d} = TF_{t,d} \times IDF_t \quad (2)$$

where,

$$TF_{t,d} = \frac{number\ of\ occurance\ of\ t\ in\ d}{total\ length\ of\ d}$$

$$IDF_t = log\Big(\frac{number\ of\ documents\ in\ P}{number\ of\ documents\ contain\ the\ term\ t}\Big)$$

2.3 Cosine-similarity

Cosine-similarity is a technique which measures the similarity between two document vectors ($\overrightarrow{d_1}$ and $\overrightarrow{d_2}$) and can be represented as follows:

$$\textit{cos-sim}(\overrightarrow{d_1}, \overrightarrow{d_2}) = \frac{\overrightarrow{d_1}.\overrightarrow{d_2}}{|\overrightarrow{d_1}| * |\overrightarrow{d_2}|}$$

2.4 Fuzzy C-Means

Fuzzy C-Means (FCM) algorithm (Bezdek et al., 1984) tries to distribute a finite collection of n documents into c clusters. It returns a list of c cluster centroids along with a matrix which shows the degree of membership of each document to different clusters. It aims to minimize the following function:

$$T_m = \sum_{i=1}^{n}\sum_{j=1}^{c} v_{ij}^{m}||d_{ij}||^2$$

where, distance $d_{ij} = x_i - c_j$, m generally set to 2 is the fuzzy coefficient, c_j is the centroid(vector) of cluster j, x_i is the i^{th} document, $v_{ij} \in [0, 1]$ is the degree of membership of x_i with respect to c_j and ($\sum_{j=1}^{c} v_{ji} = 1, i = 1, ..., n$). One can iteratively find the values of c_j and v_{ij} updated with each iteration by using equations 3 and 4.

$$c_j = \frac{\sum_{i=1}^{n} v_{ij}^{m} - x_i}{\sum_{i=1}^{n} v_{ij}^{m}} \quad (3)$$

$$v_{ij} = \frac{1}{\sum_{k=1}^{c}(\frac{||d_{ij}||}{||x_i - c_k||})^{\frac{2}{m-1}}} \quad (4)$$

3 Proposed Methodology

Step 1 *Pre-processing of Documents:*
Consider a corpus having set of classes $(C = c_1, c_2, ..., c_n)$ of documents $(D = d_1, d_2, ..., d_p)$. All documents are pre-processed which includes lexical-analysis, stop-word elimination, stemming, and index terms extraction. The term-document matrix is constructed using the vector space model, where TF-IDF value is used to measure the weight of the term t_i in its respective document d_j and is shown in the Table 1.

Table 1: Term-document matrix

	d_1	d_2	d_3	...	d_p
t_1	t_{11}	t_{12}	t_{13}	...	t_{1p}
t_2	t_{21}	t_{22}	t_{23}	...	t_{2p}
t_3	t_{31}	t_{32}	t_{33}	...	t_{3p}
.	.	.	.	...	.
.	.	.	.	...	.
.	.	.	.	...	.
t_r	t_{r1}	t_{r2}	t_{r3}	...	t_{rp}

Step 2 *Clusters formation*:
The entire corpus is clustered to generate the groups of similar terms. For this purpose, traditional Fuzzy C-means clustering algorithm (MacQueen and others, 1967) is applied on the term-document matrix of the corpus which generates 's' term-document clusters $td = \{td_1, td_2, ..., td_s\}$ having each td_i of dimension $b \times p$, where b is the number of terms and is shown in the Table 2.

Table 2: Reduce term-document matrix

	d_1	d_2	d_3	...	d_p
t_1	t_{11}	t_{12}	t_{13}	...	t_{1p}
t_2	t_{21}	t_{22}	t_{23}	...	t_{2p}
t_3	t_{31}	t_{32}	t_{33}	...	t_{3p}
.	.	.	.	...	.
.	.	.	.	...	.
.	.	.	.	...	.
t_b	t_{b1}	t_{b2}	t_{b3}	...	t_{bp}

Step 3 *Top features selection from each cluster:*
Top features are selected from each term-document cluster td_i using the following steps:

i. *Computing the cosine-similarity*:
The centroid of td_i is computed using equation 5.

$$\overrightarrow{sc_i} = \frac{\sum_{j=1}^{r} \overrightarrow{t_i}}{r} \quad (5)$$

where sc_i is the centroid of td_i. Next, the cosine-similarity score between each term $t_j \in td_i$ and sc_i is computed using equation 6.

$$cos\text{-}sim(\overrightarrow{t_j}, \overrightarrow{sc_i}) = \frac{\overrightarrow{t_j}.\overrightarrow{sc_i}}{|\overrightarrow{t_i}| * |\overrightarrow{sc_i}|} \quad (6)$$

ii. *Generating synonym list*:
Select a term $t_j \in td_i$ randomly and store its synonyms using WordNet[1] in a file $synonym\_list_{old}$ which will constitute the synonym list for t_j (example shown in Table 3). Find the common terms between $synonym\_list_{old}$ and td_i, and add them to a new synonym list called $synonym\_list_{new}$ and discard them from td_i so that the synonyms of t_j will be no longer in td_i as they are already present in the new synonym list ($synonym\_list_{new}$) of t_j. Remove the old synonym list ($synonym\_list_{old}$) of t_j. Repeat this step for the remaining word of td_i till it get exhausted.

iii. *Constructing feature vector*:
Now for every randomly selected term t_j, there is a corresponding new synonym list ($synonym\_list_{new}$) contain its synonym terms. Next from each $synonym\_list_{new}$ of t_j, select the top m% terms having high cosine-similarity scores and merge them to an array IFV which will constitute the reduced feature vector of td_i. The detail discussion are shown in Algorithm 1.

Table 3: Synonym list of terms

Term	Synonym list
Explain	account for, clarify, define, elaborate, interpret, justify,
Fast	expeditiously, fleet, hastily, hasty, quickly, mercurial, quick, rapid, speedy, snappy, swiftly, rapidly, snappily, speedily, posthaste, like a flash
File	directory, data, case, book, folder, list, information, register, repository, charts, documents, cabinet
Index	pointer, mark, needle, indicator, ratio, rule, symbol, formula, token
Program	plan, schedule, curriculum, bill, syllabus, record, timetable, bulletin, arrangements
Thesaurus	glossary, lexicon, terminology, vocabulary, reference book, source book

Step 4 *Generating the training feature vector:*
Repeat step 3 for all term document cluster td_i and merge the terms of each IFV into a final array RFV after removing all the duplicate terms. Now RFV is the required reduced training feature vector used for classification.

Step 5 *Training SVM on reduced feature vector:*
The SVM classifier is trained using the train-

[1]http://wordnet.princeton.edu/

Algorithm 1: Top feature selection

Data: Cluster td_i having cosine-similarity values of each term t_j
Result: Final feature vector (RFV) of td_i
$Term\_List(TL) \leftarrow \phi$
$Synonym\_List_{t_j}(SL_{t_j}) \leftarrow \phi$
$New\_Synonym\_List_{t_j}(NSL_{t_j}) \leftarrow \phi$
$Extra\_List(EL) \leftarrow \phi$ //A two dimensional list
$TL \leftarrow$ terms of td_i
for *each term* $t_j \in$ *TL (selected randomly)* **do**
  $SL_{t_j} \leftarrow$ all the synonyms of t_j found in Wordnet
  for *each term* $t_k \in TL$ **do**
    // except $\{t_j\}$
    if t_k *present in* SL_{t_j} **then**
      add t_k to the NSL_{t_j} of t_j and drop t_k from TL
    end
  end
  $EL \leftarrow EL \cup NSL_{t_j}$ //merge the synonym required list of t_j to EL
  $NSL_{t_j} \leftarrow \phi$
  $SL_{t_j} \leftarrow \phi$
end
for *each* $NSL_{t_j} \in EL$ **do**
  select the top m% terms T having highest cosine-similarity values from NSL_{t_j}
  $RFV \leftarrow RFV \cup T$
end
return RFV

ing feature vector (RFV) on positive definite kernels such as linear, RBF, polynomial, and sigmoid kernels. Following this, the SVM is trained using conditionally positive definite kernels such as Negative Euclidean, log Power kernels, and proposed log cosine kernel.

4 Experimental Section

Four benchmark datasets are used for experimental work (DMOZ[2], 20-Newsgroups[3], Reuters, Classic3[4] and WebKB[5]). The details of these datasets are discussed below:

4.1 20-Newsgroups

20-Newsgroups is a standard machine learning dataset and it has 11293 training and 7528 testing documents classified into 20 classes. Three classes are taken into consideration (*alt.atheism*, *soc.religion.christian* and *misc.forsale*) for experimental purpose, consisting of total 1663 training and 1107 test documents. Total number of terms used is 20422 and among them, 16270 are used for training.

4.2 DMOZ

DMOZ is one of the largest dictionaries on the Web. It has 14 categories out of which 3 categories (*Arts, Homes and Science*) of 5238 documents are used for experimental purpose. Among them, 3142 number of documents are used for training and rest are used for testing. Total number of features is 24320 out of which 19886 are considered for training.

4.3 Reuters

Reuters is a widely used text mining dataset. It has 5485 training and 2189 testing documents classified into 8 classes, where all class documents are considered for evaluation. The total number of terms used is 17582 and among them 13531 are used for training.

4.4 WebKB

WebKB is a widespread text mining dataset in which the web pages are collected from four different college websites. It has 2803 training and 1396 testing documents classified into four classes, where all class documents are considered for evaluation. The total number of terms of all these documents is 7606 and from that 7522 terms are used for training.

4.5 Classic3

Classic3 is a widespread data mining dataset. It has 4257 training and 2838 test documents classified into 4 classes: *cacm, cisi, cran, med*, having 3204, 1460, 1400, and 1033 documents respectively. All the classes are considered in the evaluation. The total vocabulary contained in all documents is 21299 and from that 15971 terms are selected for training.

We compute the test accuracy, 100-Fold cross validation accuracy, precision, recall and F-Score

[2]https://www.dmoz.org/
[3]http://qwone.com/∼jason/20Newsgroups/
[4]http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets/
[5]http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/

for aiding our comparison of various kernel techniques. Tables 4 - 8 show the detail results of classification using SVMs on different kernels for various datasets (maximum results are marked as bold in each table).

4.6 Performance Evaluation

The following parameters are used to measure the performance of the classifier.

i. *Accuracy (acc)* is the ratio between the sum of true positive cases, *TP* (number of documents that are that are classified correctly) and true negative cases, *TN* (number of documents that are not classified correctly and are not retrieved by the approach) with the total number of documents, *N = TP + FP + TN + FN*. It can be represented as follows:

$$acc = \frac{(TP + TN)}{N}$$

where,
FP: number of documents that are not classified correctly and are retrieved by the approach and *FN*: number of documents that are classified correctly and are not retrieved by the approach.

ii. *Precision (pr)* is the fraction of the retrieved documents by the classifier that are relevant.

$$pr = \frac{(relevant_{documents}) \cap (retrieved_{documents})}{retrieved_{documents}}$$

iii. *Recall(re)* is the fraction of the relevant documents that are retrieved by the classifiers.

$$re = \frac{(relevant_{documents}) \cap (retrieved_{documents})}{relevant_{documents}}$$

iv. *F-measure (F)* is the harmonic mean of *pr* and *re*.

$$F = 2 * \left(\frac{pr * re}{pr + re}\right)$$

4.7 Discussion

From the results of all the tables, it is observed that positive definite kernels (RBF, Polynomial, and Euclidean) perform poorly in classifying the text documents; while conditionally positive definite kernels (Linear, Log-Cosine (L-cos), Log-Power (L-pow) and Negative Euclidean (N-Eue) performed significantly better performance. Although the Linear kernel based SVM performs reasonably well but this might be due to the fact that the rest of the positive definite kernels other than the linear kernel are exponential in nature. This highlights an inconsistency in the classification of text documents using positive definite kernel based SVMs.

The performance of the CPDs kernels is seen to be significantly more accurate, precise, and consistent than PDs. It is observe that both the log-power and negative Euclidean based SVMs are more effective in classifying the text documents. Both the log-power and negative Euclidean kernels deliver high precision which is significantly higher than their positive definite counterparts. It is important to note that the effective performance of the proposed modified cosine similarity measure i.e. the log-cosine distance kernel (L-cos) performs well compared to other CPDs on most of the datasets. SVMs with the negative log cosine kernel works well for text document classification and it is shown in Figure 1. Training score is 1 when the number of training examples = 1000. With increase in the number of training examples, training score decreases slightly to reach 0.99. Cross-validation accuracy increases at a decreasing rate with increase in the number of training examples and it reaches to more than 90%. Similarly, SVMs with log power kernel works well for text classification. As it can be inferred from the Figure 2 that the accuracy on training set remains 1 regardless of the number of training examples. Cross-validation accuracy increases at a decreasing rate with increase in the number of training examples and it reaches to 91%.

5 Conclusion

In this paper, we assess the effectiveness of classifying text documents using support vector machines on numerous kernel functions. Experimental results on five most popular machine learning datasets show that conditionally positive definite kernels perform more consistently and accurately than positive definite kernels. We also observed that PDs that are exponential in nature perform poorly in classifying the text documents, while the non-exponential linear kernel performs reasonably well. It is also proved that the proposed modified cosine distance based log kernel (L-cos) is indeed conditionally positive definite and generates the best results in comparison to all other existing kernels. A new feature selection technique

Table 4: SVM Classification on 20-NG dataset

Kernel	Accuracy	100FAcc	Precision	Recall	F-score
RBF	37.23	40.32	28.45	25.19	26.72
Poly	35.34	35.27	30.27	25.65	27.76
Eucl	32.03	35.18	22.15	22.06	22.10
Linear	80.62	84.05	80.27	79.72	79.99
L-cos	**85.06**	**87.83**	**83.81**	**82.81**	**83.30**
L-Pow	83.55	86.45	83.72	82.53	83.12
N-Euc	83.73	86.78	83.59	82.74	83.16

Table 5: SVM Classification on DMOZ dataset

Kernel	Accuracy	100FAcc	Precision	Recall	F-score
RBF	40.73	48.35	53.22	44.56	48.52
Poly	38.43	45.24	30.71	35.35	32.86
Eucl	51.72	50.34	42.65	39.30	40.90
Linear	82.56	79.23	80.76	77.27	78.97
L-cos	**84.38**	**85.81**	**81.84**	80.88	81.35
L-Pow	79.98	81.48	80.70	**83.35**	**82.00**
N-Euc	81.71	82.47	80.52	78.27	79.37

Table 6: SVM Classification on Reuters dataset

Kernel	Accuracy	100FAcc	Precision	Recall	F-score
RBF	32.12	35.47	28.78	35.87	31.93
Poly	35.53	35.28	30.87	31.98	31.41
Eucl	22.56	38.65	42.71	38.54	40.51
Linear	79.46	80.38	76.83	74.67	75.73
L-cos	**84.68**	86.51	**81.67**	**80.45**	**81.05**
L-Pow	81.35	86.43	77.67	78.93	78.29
N-Euc	80.57	**86.76**	80.54	78.45	79.48

Table 7: SVM Classification on Classic3 dataset

Kernel	Accuracy	100FAcc	Precision	Recall	F-score
RBF	37.65	35.72	28.36	29.30	28.82
Poly	47.33	48.22	30.30	35.87	32.85
Eucl	32.06	35.13	22.15	20.08	21.06
Linear	81.63	83.98	79.24	79.87	79.55
L-cos	**88.65**	83.88	**81.88**	**83.47**	**82.66**
L-Pow	82.66	82.47	76.34	74.58	75.44
N-Euc	80.75	**84.79**	75.55	77.64	76.58

is proposed which increases the performance of the classification results. To extend the work, it is needed to prove and adopt other conditionally positive definite functions as kernels for text documents classification. Further work can include the clustering of text documents on different datasets and observe the performance of these kernels using SVM.

References

[Aggarwal and Zhai2012] Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer.

[Bezdek et al.1984] James C Bezdek, Robert Ehrlich, and William Full. 1984. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2):191–203.

Table 8: SVM Classification on WebKB dataset

Kernel	Accuracy	100FAcc	Precision	Recall	F-score
RBF	35.25	35.27	21.26	25.87	23.34
Poly	34.63	29.45	22.76	19.45	20.97
Eucl	32.05	31.14	22.67	24.32	23.46
Linear	77.26	75.22	71.73	70.98	71.35
L-cos	83.23	**86.18**	**81.53**	**83.74**	**82.62**
L-Pow	**83.27**	80.34	70.34	71.65	70.98
N-Euc	81.74	82.67	72.34	74.56	73.43

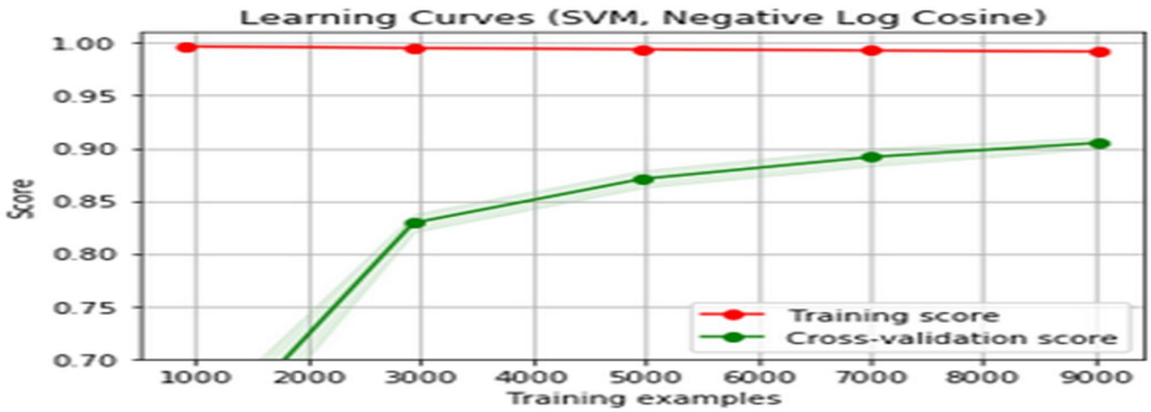


Figure 1: Training examples vs Accuracy (N-Log Cosine kernel)

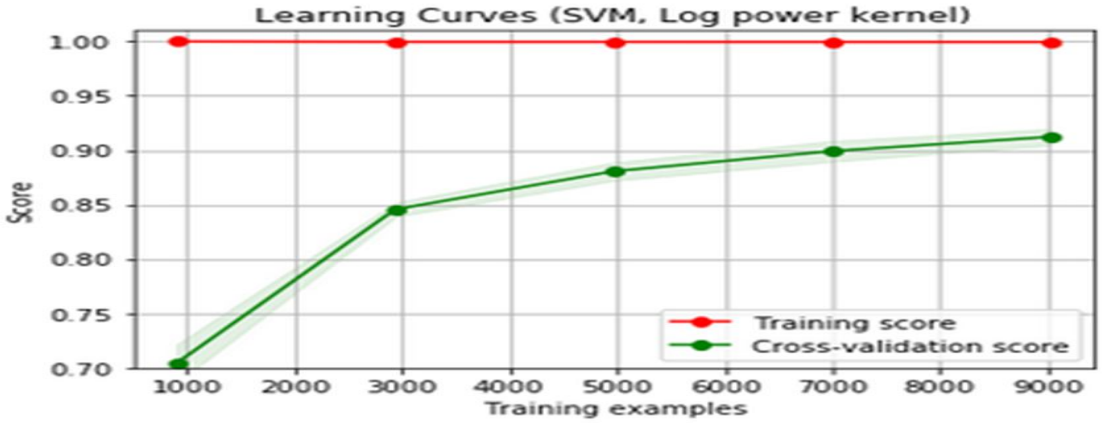


Figure 2: Training examples vs Accuracy (Log power kernel)

[Boughorbel et al.2005] Sabri Boughorbel, J-P Tarel, and Nozha Boujemaa. 2005. Conditionally positive definite kernels for svm based image recognition. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 113–116. IEEE.

[Cortes and Vapnik1995] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

[Cowling1983] Michael G Cowling. 1983. Harmonic analysis on semigroups. *Annals of Mathematics*, pages 267–283.

[Hamsici and Martinez2009] Onur C Hamsici and Aleix M Martinez. 2009. Rotation invariant kernels and their application to shape analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):1985–1999.

[Hong et al.2016] Haoyuan Hong, Biswajeet Pradhan, Dieu Tien Bui, Chong Xu, Ahmed M Youssef, and Wei Chen. 2016. Comparison of four kernel functions used in support vector machines for landslide susceptibility mapping: a case study at suichuan area (china). *Geomatics, Natural Hazards and Risk*, pages 1–26.

[Joachims1998] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. *Machine learning: ECML-98*, pages 137–142.

[Kira and Rendell1992] Kenji Kira and Larry A Rendell. 1992. The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, volume 2, pages 129–134.

[Kohavi and John1997] Ron Kohavi and George H John. 1997. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324.

[Lal et al.2006] Thomas Navin Lal, Olivier Chapelle, Jason Weston, and André Elisseeff. 2006. Embedded methods. In *Feature extraction*, pages 137–165. Springer.

[Lodhi et al.2002] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444.

[MacQueen and others1967] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[Maji et al.2013] Subhransu Maji, Alexander C Berg, and Jitendra Malik. 2013. Efficient classification for additive kernel svms. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):66–77.

[Ponte and Melko2017] Pedro Ponte and Roger G Melko. 2017. Kernel methods for interpretable machine learning of order parameters. *arXiv preprint arXiv:1704.05848*.

[Qi and Davison2009] Xiaoguang Qi and Brian D Davison. 2009. Web page classification: Features and algorithms. *ACM computing surveys (CSUR)*, 41(2):12.

[Qiu et al.2011] Xipeng Qiu, Xuanjing Huang, Zhao Liu, and Jinlong Zhou. 2011. Hierarchical text classification with latent concepts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 598–602. Association for Computational Linguistics.

[Roul and Rai2016] Rajendra Kumar Roul and Pranav Rai. 2016. A new feature selection technique combined with elm feature space for text classification. In *13th International Conference on Natural Language Processing*, pages 285–292.

[Roul and Sahoo2017] Rajendra Kumar Roul and Jajati Keshari Sahoo. 2017. Classification of research articles hierarchically: A new technique. In *Computational Intelligence in Data Mining*, pages 347–361. Springer.

[Roul et al.2016] Rajendra Kumar Roul, Aditya Bhalla, and Abhishek Srivastava. 2016. Commonality-rarity score computation: A novel feature selection technique using extended feature space of elm for text classification. In *Proceedings of the 8th annual meeting of the Forum on Information Retrieval Evaluation*, pages 37–41. ACM.

[Roul et al.2017] Rajendra Kumar Roul, Shubham Rohan Asthana, and Gaurav Kumar. 2017. Study on suitability and importance of multilayer extreme learning machine for classification of text data. *Soft Computing*, 21(15):4239–4256.

[Scholkopf2001] Bernhard Scholkopf. 2001. The kernel trick for distances. *Advances in neural information processing systems*, pages 301–307.

[Sebastiani2002] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

[Sparck Jones1972] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

[Tong and Koller2001] Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66.

[Zhang et al.2008] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. 2008. Text classification based on multi-word with support vector machine. *Knowledge-Based Systems*, 21(8):879–886.

Document Embedding Generation for Cyber-Aggressive Comment Detection using Supervised Machine Learning Approach

Shylaja S S, Abhishek Narayanan*, Abhijith Venugopal*, Abhishek Prasad*
Department of Computer Science and Engineering
PES University, Bangalore, India
shylaja.sharath@pes.edu, abhishek.1010n@gmail.com, abhijith1998@gmail.com, abhishek.pes2016@gmail.com

Abstract

Cyber-bullying may be defined as the employment of technological means for the purpose of harassing, threatening, embarrassing, or targeting a particular person. It is also possible for Cyber-bullying to have occurred accidentally. One of the major challenges in identifying cyber-bullying or cyber-aggressive comments is to detect a sender's tone in a particular text message, email or comments on social media, since what a person may consider to be a joke, may act as a hurting insult to another. Nevertheless, cyber-bullying may prove to be non-accidental in specific cases where a repetition in the pattern of text in emails, messages, and online posts is existent. In order to curb such a social threat, this Paper proposes the usage of a combination of document embeddings along with different supervised machine learning algorithms to get optimized results in flagging cyber-aggressive comments. Extensive experimentation indicates that the SVM model with rbf kernel combined with document embeddings is capable of efficiently classifying unseen test comments with an accuracy score of 88.465 % and has surpassed other models in various evaluation metrics.

1 Introduction

Social media may be thought of as interactive media which let people read and write their views. Social media lets people present their talent as it gives the user freedom to build content and share it with ease, to large groups or to the society. Hence, social media is a platform where users not only generate data, but also consume it. Hence, any person having an access to internet holds the ability to produce media contents. As social media is popular among adolescents, cyber-bullying reports are increasing day by day. Smith et al (2008), defined cyber-bullying as an aggressive and intentional behavior of an individual or a particular group using electronic forms of contact that is carried out repeatedly and over time against an individual or a certain group who cannot easily defend themselves. Going by the definition of cyber-bullying by Smith et al (2008), any behaviour showing signs of bullying on social media is also considered cyber-bullying. Also, since it occurs online and is anonymous to a certain extent, tracing such behaviour to it's source can be challenging. Hence there is a need to have an effective cyber-bullying detection system to monitor comments posted in social media and efficiently flag comments as cyber aggressive or safe.

The first step in this objective is to obtain manually labelled data for training and testing purposes, in which comments have been collected from various social networking sites and have been labelled according to whether they are cyber aggressive or not. Such labelled comments have been taken from various sources in Kaggle and Github websites. After accumulating a large number of comments from various datasets containing manually labelled comments scraped from various social media sites, they were split into datasets containing 20645 training and 8817 testing comments. Corresponding to the comments contained in the datasets, vector representations or document embeddings are generated by Doc2Vec which are subsequently fed to the supervised machine learning algorithms for training and then predicting test labels.

The organization of the remaining part of the Paper is as follows. A summarized survey of various related research experiments and literatures has

*Authors contributed equally

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 348–355

been elaborated in Section 2. The methodology proposed has been explained in Section 3. Section 4 includes the results obtained. The conclusions obtained from these results has been given in Section 5 along with a brief mention of the future work to be carried out.

2 Related Work

Despoina Chatzakou et al (2017), in an attempt to flag cyber-aggressive comments presented a method of classification by identifying behavioural aspects of cyber-bullies which differentiated their comments from others.They presented a principled and scalable approach for eliciting user, text, and network-based attributes of Twitter users, by extracting a total of 30 features and identifying the differentiating features. This paper has used word embeddings among their features.

In order to optimize detection of cyber aggressive comments, Vikas S Chavan and Shylaja S S (2015), proposed that using two additional features, simultaneously with conventional feature extraction techniques like TF-IDF and N-gram, increases the accuracy of the system up to 86% using logistic regression. This paper included two new features, which included pronoun capturing and the use of skip-grams.

Liew Choong Hon and Kasturi Dewi Varathan (2015), proposed a cyber-bullying detection system for tweets, with their focus on five types of words indicating cyber-bullying, which they deduced through their study. They used keyword matching for flagging cyber-bullying in tweets after capturing the keywords from tweets by various users.

Among other research activities carried out in the field include an effort by Kelly Reynolds and April Kontosthatis (2011), in which the data was accumulated from the Formspring.me website and labelled using Amazon's Mechanical Turk service. This labelled data was then employed to train a machine learning model to identify cyber-bullying comments through the usage of the weka toolkit.

In related text classification problems such as sentiment analysis of comments, the incorporation of paragraph vectors or document embeddings has been found to be efficient for the purpose of generating dense and low dimensional feature vectors for semantically representing entire comments or paragraphs unlike the feature matrices obtained from standard feature extraction techniques like n-grams or it's special case bag-of-words. An expedition was carried out by Parinya Sanguansat (2016) in which the employment of an unsupervised deep learning technique for numerically representing text comments in the form of document embeddings or paragraph vectors with machine learning algorithms proved to be more effective than standard methods for the task of sentiment analysis of comments on social media. Since the detection of cyber-aggressive comments is also a binary text classification task, this Paper proposes the incorporation of paragraph vectors as features to be learnt for classification by machine learning algorithms. Various classifiers are subsequently tested and evaluated to come up with an effective model for identification of cyber-aggressive comments.

3 Proposed Method

The preliminary step involved in our proposed methodology is to generate a vector sequence for each of the comments in the dataset, that represents the semantic meaning of the document or the comment, which can then be processed by machine learning algorithms to associate test data with labels. We perform extensive experimentation and evaluation on several machine learning algorithms and compare the results based on these parameters to find a suitable model which can efficiently perform the task involved.

3.1 Pre-Processing

The inability of machine learning algorithms to process raw text directly is a keen issue in the field of natural language processing. This brings out the necessity for numerical representations of linguistic units, for the purpose of which several standard feature extraction techniques such as Bag-of-Words, n-grams, etc. Though these models have been shown to be considerably effective and are the state-of-the-art models for generating vector representations for text, yet these models do not take into account the order of words in a sentence, which is an important parameter upon which the detection of cyber-aggressive comments is dependent. Also, there is a necessity for dense feature vectors of suitable dimensions unlike those provided by the bag-of-words or n-gram models which are sparse and high dimensional feature matrices. Such dense feature matrices are also ob-

tained from other models such as word2vec, which may be incorporated as well, but are more preferred in problems involving identification of analogous words or classification of topics in a sentence. In order to tackle this issue of obtaining a favourable feature matrix for the task, this Paper proposes the incorporation of document embeddings or paragraph vectors generated through Doc2Vec which is an unsupervised learning algorithm to effectively generate semantic vector representation of comments and paragraphs which fits our purpose, as we deal with multiple line comments as well. Though Doc2Vec consists of two architectures for generating paragraph vectors, namely the Distributed Bag of Words (PV-DBOW) and the Distributed Memory (PV-DM) models, the PV-DM architecture has been incorporated in our pre-processing step, not only due to the fact that it outperforms the PV-DBOW model as per the report by T Mikolov (2014) but also because it takes into account word order, leading to better results in flagging cyber-aggressive comments.

Further details include a brief summarization of the distributed memory model of Doc2Vec as used in our pre-processing step.

In the distributed memory framework, every comment or set of comments and all of the words are associated with a corresponding unique vector representation for each depicted by separate column matrices for comments and words. The model is trained such that the vector representations play a role in predicting succeeding words, taking into account various contexts which are sampled from the comments. Typically, this prediction task is performed by softmax or other multi-class classifiers. The framework performs concatenation operation for aggregating the vector representations. Generally using stochastic gradient descent, where the gradients are obtained through the back-propagation algorithm, the vectors are then trained. Therefore, since the error may be calculated at each step and be employed to upgrade the parameters of the model, the framework is capable of capturing semantics even though the vectors were initialized randomly. Using gradient descent while performing an inference step , we retrieve the vectors corresponding to a new comment or multiple lines of comments. Thus, once the weights and vectors for seen comments are obtained, the inference step helps in retrieving vectors for unseen comments as well.

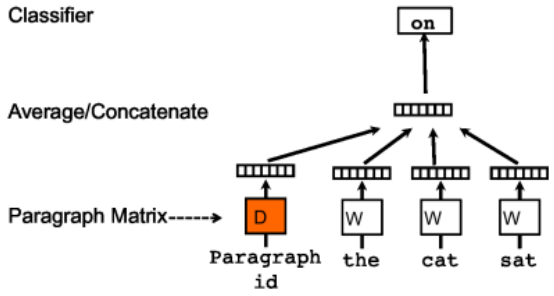


Figure 1: The above figure portrays framework for the purpose of learning paragraph vectors.This framework includes the addition of paragraph token that is mapped to a vector via matrix D. In order to predict the fourth, word the average or concatenation of this vector with a context of three words is used in this model. The paragraph vector not only represents the missing information from the current context but can also act as a memory of the topic of the paragraph. Figure adapted from the report in T Mikolov(2014)

In our experimentation, the sentences are tokenized and each set of tokens is associated with a paragraph id or tag before training, indicating the document type the sentence comes from. For convenience in our experimentation, we generate

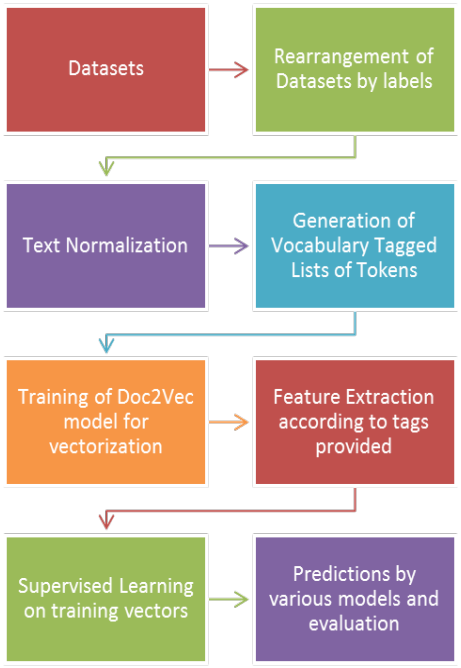


Figure 2: Methodology involved in our proposed method

the document ids with respect to the files the comments come from, because of which the training and testing data is split into a pair of files each containing cyber-aggressive and non cyber-aggressive comments. The tags are then conveniently generated with a prefix indicating whether the data is from training or testing dataset and whether it is cyber-aggressive or non cyber-aggressive. This prefix is coupled with a unique index for each comment for uniqueness and to facilitate retrieval of vectors after training. This aggregation of tagged tokenized comments from training and testing pairs of datasets are shuffled randomly for better training and eliminating any dependency on the order of feeding the input, are then fed to the Doc2Vec model for training. The training has been performed for 10 epochs in order to obtain better results. Though the number of epochs here improves the training and model performance, yet it is not a fixed parameter as such and is to be tuned according to the purpose. Typically 10 epochs is found to be sufficiently suitable for generating favourable features and therefore our experimentation includes this parameter as such. After training is performed, we extract the feature vectors for training and testing data into separate arrays with their corresponding labels in separate arrays, for being fed to machine learning algorithms for classification. The number of dimensions of the dense feature vectors has been chosen to be 100, found to be optimum, neither being too high or too low a value for being learnt by machine learning algorithms. Since the training arrays are arranged such that the cyber-aggressive and non cyber-aggressive comments are grouped together, we shuffle the arrays randomly to ensure that the models remain independent of the feeding order of the input vectors.

3.2 Classification

Following are the various Machine Learning Classification algorithms which have been trained using the document embeddings and tested for generating the predicted labels of test data :

3.2.1 Support Vector Machines (SVM)

Support vector machine, commonly referred as SVM, is one of the most common machine learning algorithm used for performing binary classification on data.It has always proved itself worthy in the field of supervised machine learning.They are motivated by the principle of optimal separation, the idea that a good classifier finds the largest gap possible between data points of different classes.Ideally, the classification boundary will be a curve or a hyper-plane that goes right down the middle of the gap between classes, because this would be the classification boundary which will have the maximum distance from the nearest data points (referred to as support vectors). This algorithm being based on the principle of optimum separation, is aimed at finding the largest distance between data points of separate classes. Ideally, the decision boundary for this classifier is a curve or a hyper-plane such that it possesses the maximum distance to the nearest data points known as support vectors. After training for classification task, an SVM is capable of efficiently predicting the class in which other data points fall, since there is only a necessity of few support vectors, due to which other data points may be neglected. We use the following three kernels for the SVM model for flagging cyber-aggressive comments.

linear kernel

$$K(X, Y) = X^T Y \tag{1}$$

polynomial kernel

$$K(X, Y) = (\gamma X^T Y + r)^d, \gamma > 0 \tag{2}$$

rbf (radial bias function) kernel

$$K(X, Y) = exp(-\gamma \parallel X - Y \parallel^2 /2\sigma^2), \gamma > 0 \tag{3}$$

Where r,d and gamma refer to the kernel parameters and K(X,Y) corresponds to the dot product of input points mapped into the feature space Y by the transformation function. However, only the results for rbf kernel have been portrayed in Section 4 since it is found to have maximum accuracy among the three kernels and it also has surpassed other kernels in various other evaluation metrics.

3.2.2 Logistic Regression

Logistic regression refers to the fitting of a linear model to the data which gives a real number. Since this number does not directly contribute to classification, it is fed into the logistic function which is :

$$\sigma = \frac{1}{1 + e^{-}x} \tag{4}$$

The sigmoid function enables the normalization of the numbers fed to be in the range 0 and 1, which facilitates the interpretation of the number obtained as a probability, which in this case is the probability of comments being cyber-aggressive.

3.2.3 Bernoulli Naive Bayes Algorithm

The Bernoulli Naive Bayes classifier uses the implementation of Naive Bayes training along with it's outstanding classification algorithms for the given data, assuming the data distribution to be a multivariate Bernoulli distribution, wherein multiple features may be included, however individual features are assumed to possess binary values. Hence it is necessary for samples to be represented as feature vectors with binary values. It is therefore necessary for the classifier implementation to binarize data before learning if the data handled is already not in the required form. Bernoulli naive Bayes's decision rule is build on :

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i) \tag{5}$$

3.2.4 Decision Trees

Decision Trees work on a sequence of test queries and answers with conditions which have been structured as a tree. In such trees, the root node and internal nodes consist of characteristic test conditions for the purpose of segregating data with different attributes. The terminal nodes of such trees possesses an assigned label which is typically a 0 or a 1. For the purpose of classification using a decision tree, the process begins at the root node where the test condition is applied to a data instance. Depending on the result of this step, the relevant branch is chosen and followed subsequently, thereby leading to either another internal node where a different test case is to be applied to decide the further path or to a terminal leaf node, where the data instance is associated with a class label.

3.2.5 Random Forest Classifier

Random Forest Classifiers are composed of set of diverse decision trees with the incorporation of randomness in their construction. The predictions of the individual classifiers are averaged to generate the prediction of the ensemble. The individual trees of such an ensemble are sampled from the training set with replacement. In these trees, when a particular node is split, the split is performed only after discovering the best way of splitting among a subset of features which are chosen randomly instead of the set of all features. Therefore, random forest classifiers possess greater bias than a single tree without randomness. However this is compensated, often in excess by the lower variance of random forests due to which an overall good model is created.

4 Results

Extensive experimentation is performed by testing and evaluating the models using the test dataset consisting of 8817 comments in all.A detailed comparison has been made by applying various evaluation metrics on the different models.

4.1 Evaluation Metrics on various Models

The various evaluation parameters that we have applied to compare the models when applied to the test dataset are as follows :

4.1.1 Accuracy score

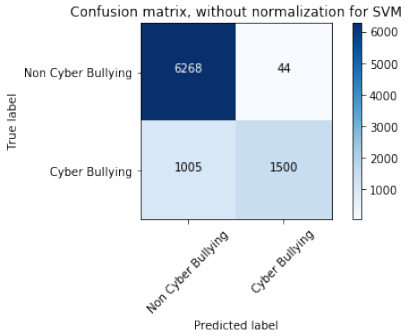
The accuracy score is a metric used in multi-label classification tasks, which corresponds to a measure of the number of data samples which have been accurately labelled according to the set of test labels provided.

4.1.2 K-Fold Cross Validation

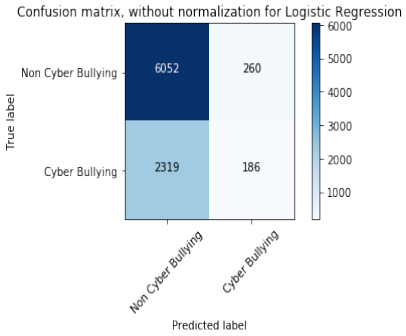
For evaluating a model using the K-fold cross-validation technique, we take the original sample and then randomly split it into equal sized k sub-samples , of which only a single sub-sample is assigned for testing purpose, which is known as the validation set. The remaining sub-samples are used for the purpose of training of the models to be evaluated. We repeat the process of cross-validation k times , taking care that each of these k sub-samples is involved in the process of testing or validation only once. In order to obtain a single value to evaluate the models, the arithmetic mean of all of the k results thus obtained is taken. The value of k has been taken as 20 in our experimentation, but in general k remains an unfixed parameter as such.

4.1.3 Confusion matrices

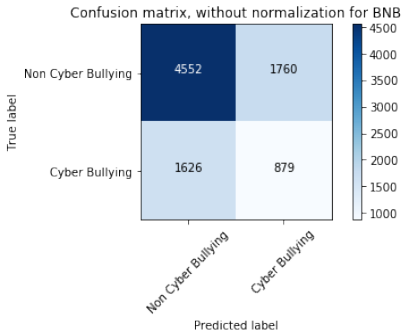
A confusion matrix is an evaluation metric which summarizes the prediction results obtained for a classification problem. It provides exact counts of the number of accurate and inaccurate predictions made by a classifier for each class. For a binary classifier, the information provided by such matrices include all four possible ways by which data may has been classified as follows :



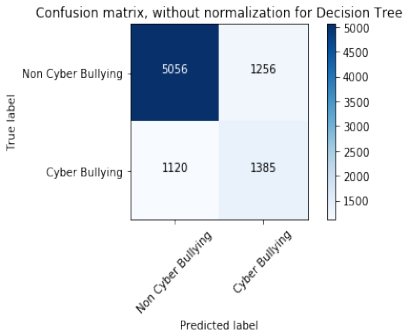
(a) SVM with rbf kernel



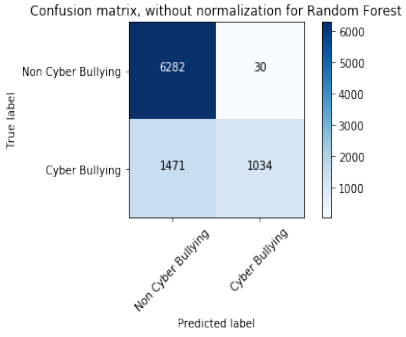
(b) Logistic Regression



(c) Bernoulli Naive Bayes



(d) Decision Tree



(e) Random Forest

Figure 3: Confusion Matrices for the various mod-

- Frequency of accurate predictions stating an instance to be negative.
- Frequency of inaccurate predictions stating an instance to be positive.
- Frequency of inaccurate predictions stating an instance to be negative.
- Frequency of accurate predictions stating an instance to be positive.

4.1.4 Precision

Precision corresponds to a measure of relevance of the results obtained from a model. It is given by.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

where TP refers to the frequency count of true positives, whereas FP is the frequency count of false positives.

4.1.5 Recall

Recall value is an evaluation metric which corresponds to a measure of how many of the results obtained from a model are actually relevant. Recall may be written as :

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

where TP corresponds to the frequency of true positive predictions while the term FNP corresponds to frequency of false negative predictions.

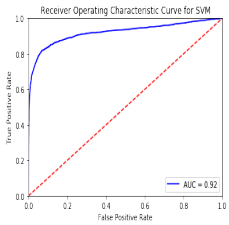
4.1.6 F-Beta-score

This metric corresponds to the weighted mean of precision and recall . The best value of this metric when evaluating a model is 1, the worst being 0. The value of beta acts as the factor which determines the weight of precision final score. A beta value less than 1 signifies that precision is weighed more whereas a beta value greater than 1 indicates recall is favoured. A beta value equal to one as used in our evaluation indicates both are weighed equally.

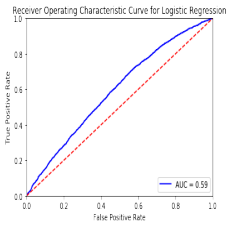
4.1.7 Area under ROC Curve

A receiver operating characteristic curve or ROC curve refers to the plot of the true positive rate or TPR values obtained against the false positive rate or FPR values obtained for the models tested at several threshold settings. The area under this

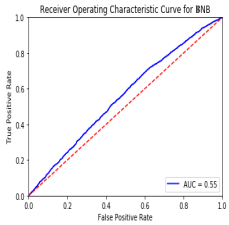
curve acts as an evaluation metric to obtain an optimum model. The best value of this score for an ideal model is 1.0.



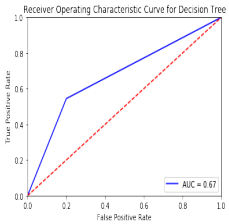
(a) SVM with rbf kernel



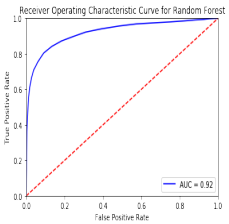
(b) Logistic Regression



(c) Bernoulli Naive Bayes



(d) Decision Tree



(e) Random Forest

Figure 4: Receiver Operating Characteristic Curves for the various modes tested

RESULTS			
Algorithm	**Accuracy Score**	**20-Fold Cross Validation Score**	**Time consumed for training and prediction (in seconds)**
SVM (rbf kernel)	88.465 %	82.179 %	1619.232
Logistic Regression	70.749 %	73.979 %	0.943
Bernoulli Naive Bayes	61.506 %	68.951 %	0.243
Random Forest	83.033 %	78.759 %	17.489
Decision Tree	72.734 %	70.666 %	13.303

Table 1: EVALUATION METRICS

RESULTS				
Algorithm	**AUROC Score**	**Precision**	**Recall**	**F-score**
SVM (rbf kernel)	0.92	0.89	0.88	0.88
Logistic Regression	0.59	0.63	0.71	0.62
Bernoulli Naive Bayes	0.55	0.62	0.62	0.62
Random Forest	0.92	0.86	0.83	0.80
Decision Tree	0.67	0.73	0.73	0.73

Table 2: EVALUATION METRICS

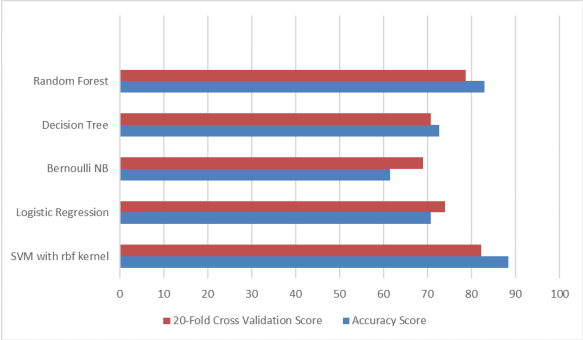


Figure 5: Comparison of the various models based on Accuracy Score and 20-Fold Cross Validation

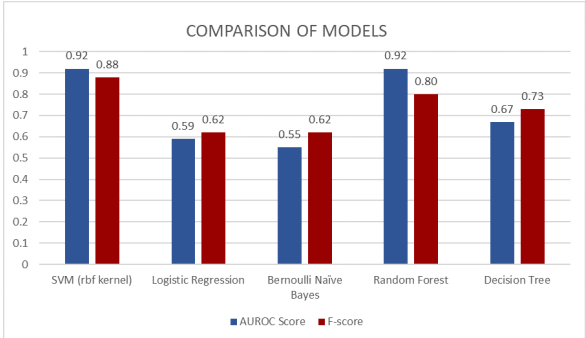


Figure 6: Comparison of the various models based on AUROC Score and F1-Score

4.2 Inference

Based on our experiments with the tested models , in labelling the test comments as cyber aggressive/non cyber aggressive, we have made a detailed summarization of various models. The metrics results have been specified in Table 1, Table 2 and in figures 3-6. Table 1 reflects the contrast between the models with respect to accuracy score, 20-fold cross validation score and time consumed for training and prediction, whereas in Table 2, we have evaluated the models based on AUROC-Score, precision, recall and f-score. Our evaluation of the tested models indicate that the highest accuracy achieved is that of the SVM model using rbf kernel, which is approximately 88.465% with an AUROC score of 0.92. Having surpassed other tested models in effectively labelling the unseen test dataset, such a model may effectively be used to flag cyber-aggressive comments which may later be used to estimate the performance of a manual based flagging system over automated approaches.

5 Conclusion and Future Work

In this Paper, we have proposed the usage of Doc2Vec to generate paragraph vectors or document embeddings as features for supervised machine learning for flagging cyber-aggressive comments. Document embeddings have been generated using Doc2Vec. We built a range of models by learning the vector representations of various comments by few supervised machine learning algorithms, and applied various evaluation metrics on the models to obtain a good efficiency in classifying comments. As a consequence of such an experiment, we found that the Doc2Vec approach coupled with SVM classifier using rbf kernel, gives an increased accuracy of approximately 88.465% in labelling test comments as cyber aggressive/non cyber- aggressive.

Further future work may be directed towards further optimization of the results obtained by applying deep learning techniques to the existing model. Further work may also be directed towards incorporating an application programming interface for real time identification of cyber-aggressive comments on social media using a model efficient in terms of both accuracy in classification as well as time taken.

References

Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. 2017. Mean birds: Detecting aggression and bullying on twitter. *arXiv preprint arXiv:1702.06877*.

Vikas S Chavan and SS Shylaja. 2015. Machine learning approach for detection of cyber-aggressive comments by peers on social media network. In *Advances in computing, communications and informatics (ICACCI), 2015 International Conference on*, pages 2354–2358. IEEE.

L Hon and K Varathan. 2015. Cyberbullying detection system on twitter. *IJABM*, 1(1).

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Radim Rehurek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer.

Kelly Reynolds, April Kontostathis, and Lynne Edwards. 2011. Using machine learning to detect cyberbullying. In *Machine learning and applications and workshops (ICMLA), 2011 10th International Conference on*, volume 2, pages 241–244. IEEE.

Parinya Sanguansat. 2016. Paragraph2vec-based sentiment analysis on social media for business in thailand. In *Knowledge and Smart Technology (KST), 2016 8th International Conference on*, pages 175–178. IEEE.

Peter K Smith, Jess Mahdavi, Manuel Carvalho, Sonja Fisher, Shanette Russell, and Neil Tippett. 2008. Cyberbullying: Its nature and impact in secondary school pupils. *Journal of child psychology and psychiatry*, 49(4):376–385.

a. For training dataset :. http://www.github.com.

b. For testing dataset :. http://www.kaggle.com.

Coarticulatory propensity in Khalkha Mongolian

Ushasi Banerjee*, Indranil Dutta* and Irfan S.**

*The EFL University
**University of Illinois at Urbana-Champaign
banerjeeushashi@gmail.com, indranil@efluniversity.ac.in, irfans2@illinois.edu

Abstract

Acoustic variation brought about by V-to V coarticulation needs to be perceptually compensated by listeners, the lack of which results in a language developing vowel harmony. This could either be mechanico-inertial where the articulatory gestures of V_1 perturb the V_2 vowel space, or anticipatory where the planning of the following vowel perturbs the space of the former. Khalkha Mongolian exhibits vowel-feature sharing of two kinds: [ATR] and Round. In this paper we examine non-harmonic sequences in Mongolian to verify coarticulatory directionality and propensity. The results show that these non-harmonic sequences exhibit a directionality and propensity which is quite the opposite of those in harmonic sequences. This finding suggests that coarticulation works as an antithetical force in non-harmonic patterns, the primary motive of which would be to maintain contrast.

1 Introduction

Continuous acoustic variation in speech is understood to be brought about by overlapping articulatory gestures (Öhman, 1966). Vowel-to-Vowel (V-to-V) coarticulation is a special case of non-contiguous coarticulation which helps model acoustic patterns resulting from anticipatory articulatory planning on the one hand, and carryover coarticulation on the other. Acoustic variation in speech could also be attributed to vocal tract shape differences, size, shape, and density of the segmental inventory (Manuel, 1990; Manuel, 1999). A uniform way of understanding the variation has been to locate it in the dynamic and kinematic differences in articulatory overlap. These differences have been modelled variously in space and time (Kirchhoff and Bilmes, 1999; Deng et al., 2006). In V_1CV_2 sequences, the standard assumption has been that the acoustics of V_1 is perturbed by the anticipatory production planning of V_2 gestures, while the mechanico-inertial properties of articulatory gestures of V_1 are known to be responsible for the acoustic perturbation of V_2. Coarticulatory acoustic variation in V-to-V sequences once phonologized are known to provide the conditioning for the development of vowel harmony patterns (Przezdziecki, 2000; Ohala, 1994). Typically, in non-harmonic languages, acoustic variation resulting from coarticulation is perceptually compensated by listeners. Lack of such perceptual compensation has been shown to be responsible for the development of vowel harmony patterns (Beddor et al., 2002).

Coarticulatory propensity is the spatial and temporal extent of coarticulation between or across segments. Broadly speaking, coarticulatory resistance or the intrinsic resistance to coarticulation that segments with variable articulatory gestures possess has also been shown to affect acoustic variation (Martin and Bunnell, 1981; Recasens, 1984). Nature and extent of coarticulation is also governed by language specific properties such as syllable structure (Manuel, 1999), coarticulatory resistance of the intervening consonant in V-to-V segment sequences (Recasens et al., 1995). In addition, the nature of contrast between segments has also been shown to affect the magnitude of coarticulation in V-to-V sequences, especially in languages that exhibit vowel harmony of the type where distinctive features are shared between vowels (Dutta et al., 2017) and in those where vowel harmony results in vowel copying (Dutta et al., 2016).

While there has been a substantial amount of work on V-to-V coarticulation in both harmony and non-harmony languages, in this paper we examine the magnitude and extent of V-to-V coarticulation in Khalkha Mongolian. Khalkha Mongolian exhibits both Advanced Tongue Root [ATR]

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 356–361

harmony and rounding harmony. Unlike, suffixal harmony systems like Assamese (Mahanta, 2008), Bengali (Shamim, 2011), and Telugu (Wilkinson, 1974; Kissock, 2009) where the direction of the harmony is right-to-left, Khalkha, primarily, but not exclusively is a stem internal harmony system where the direction of the harmony proceeds from left-to-right. Specifically, we examine the effect of formants from V_1 on V_2 and vice-versa using a set of linear fixed effect model. We observe that the direction of coarticulation in non-harmonic sequences is greater in the anticipatory direction which is opposite to the direction of suffixal vowel harmony in Khalkha Mongolian. We also find this effect only in the F_1 values and not in the F_2, which suggests that the coarticulatory formant perturbation runs in the tongue height dimension and not in the tongue front-back dimension. In section 2, we present a detailed phonological account of the vowel harmony system in Khalkha Mongolian. Following that in section 3, we provide details of our experimental methodology and speech materials. In section 4, we present the primary findings of our study, and we conclude in section 5 by motivating the need to understand the nature of vowel contrasts that mitigate the coarticulatory propensity and directionality, especially in vowel harmony languages.

2 Vowel Harmony in Khalkha Mongolian

Vowel harmony is a phonological process that restricts the co-occurrence of vowels (usually) within a non-compound word. The Mongolian vowel inventory is unevenly distributed in features, consisting of seven vowels *i, e, a, o, ɔ, u and ʊ*. These can be classified as pharyngeal: *i, e, u, o* and non-pharyngeal:*a,ʊ, ɔ*. As can be noticed, *i* does not have a pharyngeal counterpart. We find two types of vowel harmony in Khalkha Mongolian : [ATR] and Rounding. It affects roots as well as derivational/inflectional suffixes. While the initial vowel is assumed to be specified, the non-initial vowels can be underlyingly /i/, /E/, or /U/. The archiphonemes /U/ and /E/ have missing features that are filled in through vowel harmony. /U/ undergoes just [ATR] harmony, becoming either /u/ or /ʊ/, while, /E/ undergoes both [ATR] and [round] harmony, becoming /e/, /a/, /o/, or /ɔ/ (Godfrey, 2012).

2.1 [ATR] Harmony

ATR Harmony distinguishes tense and lax vowels in Mongolian, which Svantesson (2005) refers to as 'pharyngeal' and 'non-pharyngeal'. Vowels in non-compound words must share values for [+/-ATR], depending on the root/stem vowel (at the morpho-phonemic level).

- Trigger vowels: *o, ɔ, u, ʊ, a, e* (in initial positions)
- Target vowels: Archiphonemes /E/ and /U/

Alternations:

- /E/: /e/, /a/, /o/, or /ɔ/
- /U/: /u/, or /ʊ/

The vowel *i* in the initial syllable also forces the following vowels in the non-compound word to be [+ATR] (Svantesson, 1985).

In Nevins (2010) and Godfrey (2012), the harmony is conceptualized as a search-copy mechanism by 'needy' vowels instead of there being harmony 'trigger's. In [ATR] harmony, the search proceeds leftward and looks for the nearest contrastive instance of [ATR]. Once found, the value is copied. If none is found, default [+ATR] is inserted.

2.2 Rounding Harmony

This phonological process influences vowels to surface as rounded when the neighbouring vowel (the root/stem vowel for Khalkha Mongolian) is rounded. However, in most cases, conditions referring to tongue body position (height and/or backness) are imposed on either the triggering element, the target, or both (Kaun, 1995).

In Khalkha Mongolian, we observe two conditions for rounding harmony:

- *The trigger must be nonhigh.*
- *The trigger and target must agree in height.*

This kind of a system is similar to one seen in Sibe, a Tungusic language of China (Li., 1996).

- Trigger vowels: /o/, and /ɔ/
- Target vowels: Archiphoneme /E/

The archiphoneme /E/ surfaces as open rounded vowels *o, or ɔ* in the non-initial syllable when preceded by the same vowel. An open vowel that follows a non-open rounded vowel (*u, u*) must be unrounded (*e or a*) (Svantesson, 1985)

2.3 Transparent *i*

Transparent vowels are those vowels that may intervene between the trigger and the target of harmony even when they bear the opposite value for the harmonizing feature (Benus, 2010). Non-initial *i* in Mongolian is transparent, i.e., it is completely ignored by vowel harmony; neither does it participate in vowel harmony, nor does it block the process. *i* is the only vowel phoneme that is fully specified in non-initial vowels.

Example:

/po:r-ig-**E**/ po:r-ig-o *po:r-ig-e
(gloss) 'kidney-**ACC-RFL**'

In Benus (2010), phonetic and phonological investigation of transparent vowels under a dynamic model show that transparent vowels are in fact integral parts of harmonic domains. The phonetic properties of transparent vowels get integrated over phonological selection of suffixal vowels.

2.4 Opaque *u and* ʊ

Opaque vowels, in contrast, require a local agreement relationship between the trigger and the target, i.e. there can be no intervening vowel (Benus, 2010). In Khalkha, intervening non-open velar vowels block rounding harmony. Not only are they not affected by vowel harmony, they also prevent rounding harmony to spread across them.

Example:

/ɔr-ʊɮ-ɮ**E**/ ɔr-ʊɮ-ɮa *ɔr-ʊɮ-ɮɔ
(gloss) 'enter-**CAUS-DPST**'

This opacity, however, is restricted to rounding harmony; these segments don't behave so in the process of [ATR] harmony (Godfrey, 2012). Phonetic factors have also been implicated in grounding the phenomenon of opacity in 'tongue root harmony systems (Archangeli and Pulleyblank, 1994).

3 Materials and methods

Four female native Khalkha Mongolian speakers were recorded at The EFL University. The material block consisted of fifty-nine target words and six distractors in carrier phrases of the type "pi <target word/distractor> gesen". Four repetitions of each block were recorded for all speakers with a five minute break within each block. Recordings were conducted in a quiet environment. The total number of critical items was 59*4*4=944. The data was presented to the speakers in Cyrillic script. The recorded speech was segmented and annotated manually in Praat (Boersma and Weenink, 2009).

Ten vowel formants from each V_1CV_2 sequence were extracted using a Praat script (Boersma and Weenink, 2009) FormantPro (Xu, 2007 2015). Extracted formants where normalized by using the Lobanov method to eliminate specific speaker effects (Lobanov, 1971).

$$F^N_{n[V]} = \frac{(F_{n[V]} - MEAN_n)}{S_n} \quad (1)$$

Where $F^N_{n[V]}$ is the normalized value for formant n of vowel V. $MEAN_n$ is the mean value for formant n for the speaker in question and S_n is the standard deviation for the speaker's formant n.

Linear Mixed Effects model from the lme4 package in R (Bates et al., 2015) was implemented on the formant data to ascertain the effects of the distal and proximal formants on coarticulation. Distal formant data consisted of F_1 and F_2 data from vowel midpoints representing the steady-state formants where the V-to-V coarticulation effects are most distal from the formant perturbations due to the intervening consonant. Proximal formant data consisted of F_1 and F_2 data from vowel offsets of V_1 and vowel onsets of V_2.

4 Statistical analyses and results

Vowel F_1 and F_2 plots on an X-Y euclidean plane where plotted using the PhonR package (McCloy, 2016). The formant data visualizations provided information about the relative positions of the vowels and the areas of the vowel polygons for V_1 and V_2. In Fig.1 below the left panel represents the steady-state positions and the vowel area polygons for V_1 and the right panel represents the same for V_2. We make two observations with respect to the relative positions and the polygonal areas. The mid-front vowel *e* is raised and is overlapped significantly with the high front vowel *i*, while the *e* is lower than the *i*. This pattern suggests that the F_1 and F_2 values of *e* are perturbed by the presence of a high vowel in the V_2 position, which in turn implies a stronger anticipatory effect, at least for the vowel *e*.

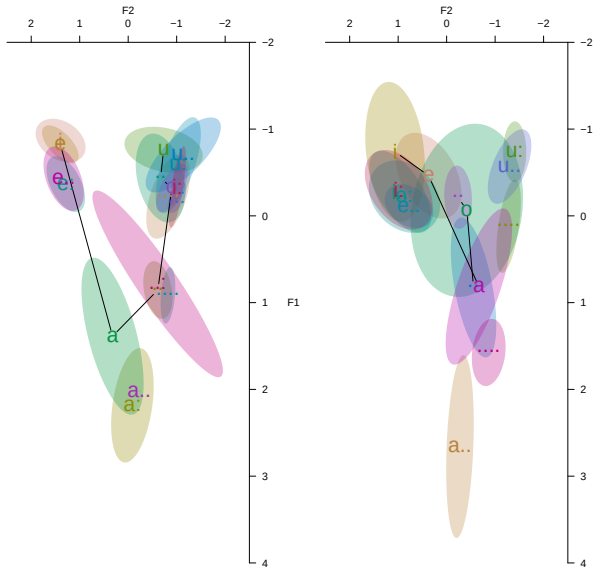


Figure 1: Steady state distal formants: Vowel ellipses and polygons from V_1T6, i.e., V_1 mid-point (Left panel) and V_2T_5; i.e.,V_2 mid-point (Right panel)

In Fig.2, the left panel represents proximal positions and vowel area polygons for V_1 and the right panel represents those for V_2. It shows that in spite of more overlapping effect due to the presence of a consonant, the polygon area of V_1 is still larger than that of V_2. From this, we make the observation that the effect of V_2 on V_1 is robust across their different positions in time (vowel-mid position, onset and offset). In Table 1., we present the vowel polygon areas from these models (distal and proximal). The values show that for all subjects, the vowel polygon area for V_1 is larger in comparison to that of V_2.

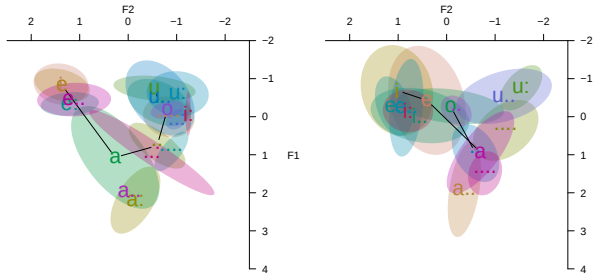


Figure 2: Proximal formants: Vowel ellipses and polygons from V_1T6, i.e., V_1 mid-point (Left panel) and V_2T_5; i.e.,V_2 mid-point (Right panel)

We present results from a linear mixed effects model which tested the variable and mutual effect of F_1 and F_2 values from distal and proximal V positions in order to quantify the coarticulatory propensity in non-harmonic V_1CV_2 sequences.

Polygon Area	Subject f1	Subject f2	Subject f3	Subject f4
Distal				
V1 at T6	**3.48591494**	**4.1172744**	**2.5307905**	**2.8948019**
V2 at T5	0.09251035	0.6124816	0.1368078	0.3408551
Proximal				
V1 at T10	**2.725822**	**3.827627**	**1.998709788**	**2.1165014**
V2 at T1	0.3790905	0.536976	0.008869308	0.1819477

Table 1: Vowel polygon areas of V_1 and V_2 from distal and proximal formant measures

In the first model, we test the effect of F_1 from T5 of V_2 and a fixed effect of V_2 on the F_1 from V_1 at T6. This model returns a t=3.543 with no significant effect of V_2. In the second model, we test the effect of F_1 from T6 of V_1 and a fixed effect of V_1 on the F_1 from V_2 at T5. This model returns a t=3.233 and a t=2.185 for vowel V_1, /u:/. Here Subject and Item function as random effects. Our results indicate that there is significant covariation of F_1 values with slightly greater effect of V_2 on V_1 suggesting that the directionality of coarticulation as seen in the F_1 values is in the anticipatory direction, which is opposite to the direction of the suffixal harmony system in Khalkha Mongolian. Similar models on F_2 do not show significant effects, with t=-0.044 and t=-0.020 for F_2 of V_1 at T6 and F_2 of V_2 at T5, respectively.

5 Contrast and coarticulatory propensity in vowel harmony systems

Stem-internal and stem + suffix harmonic sequences in Mongolian exhibit a left-right directionality. Lack of perceptual compensation (Beddor et al., 2002) is known to contribute to the development of vowel harmony systems, to the extent that it might regress to a radical case of vowel-copying, such as in Telugu (Dutta et al., 2016; Kissock, 2009; Sailaja, 1999). In this paper, we argue that in non-harmonic Khalkha Mongolian sequences the coarticulatory propensity is greater in the anticipatory direction, opposite to the direction of both ATR and rounding harmony, in an effort to maintain contrast, where harmonic sequences may lead to contrast obliteration in terms of featural neutralization. The directionality of coarticulatory propensity, also suggests that articulatory planing seeks to preserve contrast. Unlike the intuitive notion of coarticulation as a force that contrives against contrast, Mongolian shows that it might also be one that does not, and indeed it aug-

ments the force of contrast. Coarticulation could be viewed as a force that is not merely functioning at the production end of language but also at the planning end. In vowel harmony languages, where lack of perceptual compensation for coarticulatory acoustic variation may have lead to the development of complex feature sharing in V-to-V contexts, the V-to-V coarticulation patterns in non-harmonic sequences seek to enhance contrast by providing perceptual advantage through variable coarticulatory propensity.

References

D. Archangeli and D. Pulleyblank. 1994. *Grounded Phonology*. Current Studies in Linguistics. MIT Press.

Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. 2015. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48.

Patrice Speeter Beddor, James D. Harnsberger, and Stephanie Lindemann. 2002. Language-specific patterns of vowel-to-vowel coarticulation: acoustic structures and their perceptual correlates. *Journal of Phonetics*, 30(4):591 – 627.

S. Benus. 2010. *Dynamics and Transparency in Vowel Harmony*. Universal Publishers.

Paul Boersma and David Weenink. 2009. Praat: doing phonetics by computer (version 5.1.13).

Li Deng, Dong Yu, and Alex Acero. 2006. A bidirectional target-filtering model of speech coarticulation and reduction: Two-stage implementation for phonetic recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(1):256–265.

Indranil Dutta, S Irfan, and KRS Harsha. 2016. Using ANNs for vowel identification from V-to-V coarticulation in non-harmonic VCV sequences. Presented at the 15th Conference on Laboratory Phonology, Ithaca, NY.

Indranil Dutta, Irfan S., Pamir Gogoi, and Priyankoo Sarmah. 2017. Nature of contrast and coarticulation: Evidence from Mizo tones and Assamese vowel harmony. In *Proceedings of Interspeech 2017*, pages 224–228.

Ross Godfrey. 2012. Opaque intervention in Khalkha Mongolian vowel harmony: A contrastive account. Technical report, McGill Working Papers in Linguistics, Volume 22.1,Winter 2012.

Abigail R. Kaun. 1995. *The Typology of Rounding Harmony: An Optimality Theoretic Approach*. Cambridge University Press.

Katrin Kirchhoff and Jeff A Bilmes. 1999. Statistical acoustic indications of coarticulation. In *Proc. ICPhS*, volume 99. Citeseer.

Dworak Kissock. 2009. Telugu vowel assimilation: Harmony, umlaut, or neither? In *Proceedings of the Seventeenth Manchester Phonology Meeting*.

Bing Li. 1996. *Tungusic vowel harmony: description and analysis*. Holland, Institute of Generative Linguistics, Amsterdam.

B. M. Lobanov. 1971. Classification of Russian vowels spoken by different speakers. *The Journal of the Acoustical Society of America*, 49(2B):606–608.

Shakuntala Mahanta. 2008. *Directionality and locality in vowel harmony: With special reference to vowel harmony in Assamese*. Netherlands Graduate School of Linguistics.

S. Manuel. 1990. The role of contrast in limiting vowel-to-vowel coarticulation indifferent languages. *Journal of the Acoustical Society of America*, 88:1286–1298.

S. Manuel. 1999. Cross-language studies: Relating language-particular coarticulation patterns to other language-particular facts. In W. Hardcastle and N. Hewlett, editors, *Coarticulation: Theory, data and techniques*. Cambridge University Press.

James G. Martin and H. Timothy Bunnell. 1981. Perception of anticipatory coarticulation effects. *The Journal of the Acoustical Society of America*, 69(2):559–567.

Daniel R. McCloy. 2016. Tools for phoneticians and phonologists phonR. R package version 1.14.4.

Andrew Nevins. 2010. *Locality in vowel harmony*, volume 55. MIT Press.

John J Ohala. 1994. Towards a universal, phonetically-based, theory of vowel harmony. In *Third International Conference on Spoken Language Processing*.

Sven EG Öhman. 1966. Coarticulation in vcv utterances: Spectrographic measurements. *The Journal of the Acoustical Society of America*, 39(1):151–168.

Marek Przezdziecki. 2000. Vowel harmony and vowel-to-vowel coarticulation in three dialects of Yoruba. *Working Papers of the Cornell Phonetics Laboratory*, 13:105–124.

Daniel Recasens, Jordi Fontdevila, and Maria Dolors Pallarès. 1995. Velarization degree and coarticulatory resistance for /i/ in Catalan and German. *Journal of Phonetics*, 23(1–2):37 – 52.

Daniel Recasens. 1984. Voweltovowel coarticulation in catalan vcv sequences. *The Journal of the Acoustical Society of America*, 76(6):1624–1635.

P Sailaja. 1999. Syllable structure of Telugu. 4:743–746.

Ahmed Shamim. 2011. A reanalysis of Bengali vowel assimilation with special attention to metaphony. Master's thesis, Graduate Center, City University of New York.

Jan-Olof Svantesson. 1985. Vowel harmony shift in Mongolian. *Lingua*, 67(1985):283–327.

Jan-Olof Svantesson. 2005. *The Phonology of Mongolian*. Oxford University Press Inc, New York.

Robert W Wilkinson. 1974. Tense/lax vowel harmony in Telugu: the influence of derived contrast on rule application. *Linguistic Inquiry*, pages 251–270.

Yi Xu. 2007–2015. Formantpro.praat.

Developing Lexicon and Classifier for Personality Identification in Texts

[1]Kumar Gourav Das **[2]Dipankar Das**
[1]Department of Computer Science and Engineering
Future Institute of Engineering & Management, Kolkata, India
[2]Department of Computer Science and Engineering
Jadavpur University, Kolkata, India
[1]kumargouravdas18@gmail.com, [2]dipankar.dipnil2005@gmail.com

Abstract

Personality, an essential foundation of human behavior is difficult to identify and classify from texts because of the scarcity of explicit textual clues. Several works were attempted for personality identification by employing well-known lexicons like WordNet, SentiWordNet, SenticNet etc. However, a lexicon solely devoted for identifying different types of personality is rare. Thus, in the present article, we have discussed the methodologies to develop a personality lexicon from the Essay dataset, a personality corpus based on Big Five model. We have used a frequency based N-gram approach to extract the unique words as well as phrases with respect to each of the Big Five personality classes. In addition to the words, we have added another feature, corpus based probability of occurrence into the lexicon. Finally, we have evaluated our lexicon on a small Youtube personality dataset and found satisfactory coverage. In addition, we have developed a LIWC based classification framework by employing several machine learning algorithms followed by feature selection using information gain and correlation techniques. SVM and Logistic Regression achieved the maximum accuracies of 78.52% and 62.26% with a reduced set of feature size 15 and 10 selected by information gain and correlation attribute evaluation, respectively.

1 Introduction

Personality refers to the individual differences in characteristic patterns of thinking, feeling and behaving. Personality is considered as the most difficult human attribute to understand. Personality traits are traditionally measured through the use of questionnaires such as the Big Five Inventory (BFI) (Tett and Rothstein ,1991). However, an alternative approach is to analyze an individual's linguistic differences. Personality of a person is reflected in his behavior and speech which indirectly affects the job performance, one's effectiveness in work. Not only in jobs, there are so many other applications where we can use the advantage of personality identification including social network analysis e.g., Twitter (Pratama and Sarno, 2015), Facebook (Golbeck and Turner ,2011) (Alam Firoj and Ricardi, 2013) (Iacobelli Culotta, 2013), recommendation systems (Golbeck and Turner, 2013), sentiment analysis/opinion mining, Author Profiling (Rangel Pardo and Daelemans,2015), construction of emotion lexicon (B.G. Patra et al,2013)and many others. Personality is correlated with many other aspects of our daily life such as job success (R.P. Tett et al,1991), marital happiness (E.L. Kelly et al, 1987) too. Now, the recent trend is automatic identification of personality from some text or audio or may be video also. We can identify personality from various single modes (audio, video, texts etc.) as well as in multimodal way.

However, due to scarcity of proper audio dataset based on personality, we have started our experiment only on text dataset. We have used two standard text dataset, Essay (Pennebaker,et al. 1999) and Youtube (J.I. Biel et al., 2013).

Personality research is being nurtured as a developing field and only few works have been done till date. There are lexicons like SentiWordNet 3.0 (S. Baccianella et al, 2010), LIWC (Y.R.Tausczik et al, 2010) (F. Mairesse et al,

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 362–372

2007), Senticnet 3.0 (Cambria et al, 2012) etc. which help in identifying personality. However, to the best of our knowledge, there is no open source lexicon that contains words/phrases of a particular type of personality. Thus, one of our prime motivations is to develop lexicons for each of the Big Five personality type, separately.

In the present work, we have classified personality obtained from the written text based on the model of Big Five personality classes. The Big Five personality model is considered as a standard model for personality traits. This Big Five personality model has been used in many personality detection research works as they help in developing several applications.

According to Big Five model, personality is assessed in five dimensions of OCEAN –

a. **O**penness **(***inventive, curious***)**
b. **C**onscientiousness **(***organized***,** *efficient, sincere***)**
c. **E**xtroversion **(***energetic***,** *sociable*)
d. **A**greeableness **(***friendly, trustable and compassionate***)**
e. **N**euroticism **(***apprehensive, sensible***)**

In the present work, we have developed a lexicon of words and phrases corresponding to each of the Big Five personality classes. However, we have restricted ourselves to find only those words that belong to only one particular class of personality and not in any other class. The approach used in this work is fully automated and no manual or human interaction has been carried out. The hypothesis considered is a two tier filtration strategy; first, we identified the distinct words of each personality class that do not belong to any other class by using the set disjoint operations. Using this approach, we have obtained four different sets of words and phrases corresponding to each of the Big Five personality classes. Thereafter, we have considered the intersection of four different set of words and phrases as previously obtained and formed a lexicon for each personality class. We have used a n-gram method where in case of unigrams, we have obtained unique set of words and in case of bigrams and tri-grams, we extracted a unique set of phrases. Finally, the probability of each word or phrase has been calculated in order to add the occurrence probability information into the lexicon. In addition, we have explored the LIWC tool and developed a classification module for identifying and classifying the instances of both **Essay** dataset and **Youtube** personality dataset with a reduced set of features identified using information gain and correlation based techniques. The rest of paper is organized as follows. In Section 2, we have discussed the related work ,in Section 3 we have discussed about the dataset and preprocessing . Section 4 describes the lexicon development whereas Section 5 describes the developmental phases of LIWC based classification module. Finally, Section 6 mentions the observa tions and comparisons followed by conclusions and future work.

2 Related Work

Correlation between linguistic clues and personality traits have been identified to discover the way for carrying research in the area of automatic personality classification. We mainly focused on classifying personality traits based on text due to scarcity of multimodal dataset. To the best of our knowledge, the field be in its infancy. Though several researchers have started their struggles in identifying personality from text by adopting various approaches, n-gram always has a huge impact in most of the cases (J.Oberlander et al, 2006).

We extracted linguistic features from essay dataset using a text analysis tool, Linguistic Inquiry and Word Count (LIWC), (F.Mairesse et al. 2007), (G.Sidorov2006). Several authors used the LIWC tool for identifying the impacts of different linguistic features on different personalities as discussed in (Yla R.Tausczik et al, 2009), (F. Mairesse et al, 2007). LIWC is a text analysis tool that counts and sorts words based on their psychological and linguistic category. NRC is another lexicon that contains more than 14000 distinct words annotated with 6 emotions like *anger, fear, sadness, joy, disgust* and *surprise* along with two types of sentiments like *positive* and *negative*. The NRC lexicon has been used in other related work on personality where the authors explored the features of NRC and LIWC both (Mohammad et al., 2013) (G. Farnadi et al, 2014). MRC is a psycholinguistic database that contains psychological and distributional information of more than 150,00 words annotated with 14 features like phonemes (*Nphon*), syllables (*Nsyl*)(Coltheart, 1981) .

On the other hand, rough set based machine learning techniques have been used for personality identification (Gupta et.al 2013) whereas Naïve Bayes, KNN and SVM were also employed for

personality identification on Twitter texts (B. Y. Pratama et al. , 2015). A few authors have also investigated the age and gender related information from formal texts (Burger, J.D, 2011).

In contrast to such previous attempts, in the present work, we aimed to develop a personality lexicon of five different Big Five classes where the words even phrases are categorized according to the Big Five personality model. It has to be mentioned that one of our strict criteria that has been followed here is that no word or phrase of a particular personality class should mingle with words of other personality class. The words are also associated with their probability scores which make the lexicon useful for classifying personality from texts. Moreover, we have used information gain and co-relation techniques to conduct the feature ablation study for developing a personality classifier also.

3 Dataset and Preprocessing

In order to start with our experiments, we have used two text datasets. For developmental purpose, we have used the Essay dataset (Pennebaker, J. W., 2007) and for testing the coverage and performance evaluation purpose, we have used the YouTube dataset (J.I. Biel, 2013). Huge number of researchers used these two datasets to develop and test various personality detection models. Thus, we have considered these two as our gold standard datasets.

3.1 Eassy Dataset

Essay dataset (Pennebaker, J. W., 2007) is a large dataset that consists of 2468 text documents labeled with personality classes. The labeled personalities are based on the classes of Big Five personality traits. The classes are *Openness* (O), *Conscientiousness*(C), *Extraversion* (E), *Agreeableness* (A) and *Neuroticism* (N).

3.2 Youtube Dataset

Youtube personality dataset (J.I. Biel, 2013) consists of a collection of speech transcriptions, and personality impression scores of 404 YouTube users. These files are also tagged with the Big Five personality classes. Their speeches were transcribed by professional annotators and the transcriptions contains approximately 10K unique words and 250K word tokens.

3.3. Preprocessing Text

3.3.1. Labeling

We have started our experiments by considering each and every personality class separately because we were trying to find out unique words or phrases with respect to each of the Big Five personality classes. For that very reason, at first, we considered only those files that belong to only one specific class. Each character of such a tuple of five represents each of the Big Five personality classes (e.g.**O**penness -Y, **C**onscientiousness -N, **E**xtroversion-N,**A**greeableness-Y,**N**euroticism-N) represents the instance belongs to *Openness* and *Agreeableness*). The basic steps of pre-processing are mentioned below.

3.3.2. Lower case conversion

Change the whole text into lower case so as to maintain consistency in our further approaches.

3.3.3. Tokenizing

Change each of the sentences into a collection of single words.

3.3.4. Filtering

We have eliminated the stop words and numbers because stop words are common words that have no meaning but are compulsory to maintain the grammatical structure of language (e.g., *is, am are*). At first, we find out the count, i.e. the number of texts that belong to only one specific personality class and then the total number of texts that belong to that specific class irrespective of whether the file belongs to other classes or not. Then, we count the total number of phrases and words for both these two types of classes and calculated the percentage of occurrence of phrases and words in one specific personality class.

4 Lexicon Developing Module

We assumed that the words people use in their daily life reveals important aspects of their social and psychological uniqueness. Our objective is to explore different methods to find out words that are commonly used by the people belonging to a particular personality class. Therefore, we designed the n-gram module to identify the words or phrases that distinguishingly classify an instance of that particular class.

4.1 N-gram Module

We developed a lexicon for different personality classes that contains not only unigrams but bigrams and trigrams also. The distinctions between linguistic styles and linguistic contents can be seen in how two people may make a simple request. E.g., "*Would it be possible for you to give me a glass of water?*" and "*Give me a glass of water*" both the sentences express the speaker's desire for water and direct the listener's action. However, the two utterances also reveal the speaker's personality. N-gram feature would help us to find the unique words of each and individual personality type. Thus, in order to find the unique words of each personality class, we carried out different levels of experiment. We try to find out those texts that belong to a specific personality class using Equation 1.

$$T_{c=\ \cap_{i=1}^{n}(\theta c - \theta_i^{\circ})\ i \in all\ class\ except\ C}^{w} \quad (1)$$

$$where\ T_w^c = Total\ number\ of\ unique\ words\ of\ a\ specific\ class$$

$$\theta c = words\ belong\ to\ a\ specific\ class$$

$$\theta_i^{\circ} = words\ belong\ to\ the\ reamining$$

Fig 1: Equation for unique word count

Consequently, the frequencies of those unique words have been estimated. It was observed that stop words do not help in detecting the personality. Thus, the stop words were removed for counting the unigrams only but, for bigrams and trigrams, the stop words were not removed as bigrams and trigrams were considered to be our potential repositories of personality phrases. Initially, we estimated top 300 n-grams for each class. Then, using equation 1, we calculated the n-grams that belong to only that class. Next, the same process is repeated for obtaining top 500 and 1000 n-grams. Similarly, the unique words of other personality classes were also calculated.

E.g., a few instances of the lexicon formed for each of the Big Five personality classes along with their frequencies are "*strange"* that occurs in **openness** class 18 times, "*suppose"* that occurs in **agreeableness** class 14 times. The bigram "*really don't"* occurs in **Neuroticism** class 32 times. The trigram "*I don't know* occurs" in **Extrovert** class 40 times etc.

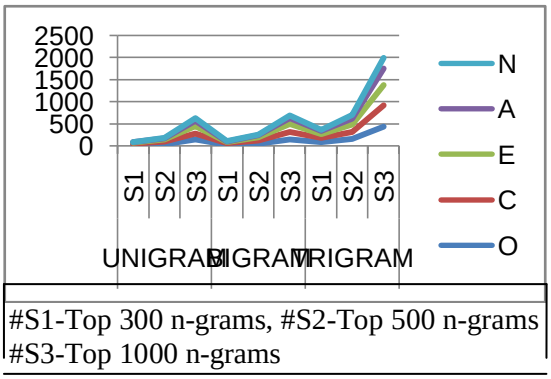


Fig 2: N-gram counts

4.2 Probability Calculation Module

The lexicon for each of the Big Five personality classes has been prepared in our previous step. Next, we need to find the occurrence in terms of probability of each word based on Equation 2.

$$P_w = \frac{T_u^c}{C_w} \quad (2)$$

$$where\ , P_w = Probability\ of\ occurence\ of\ a\ word\ in\ that\ class$$

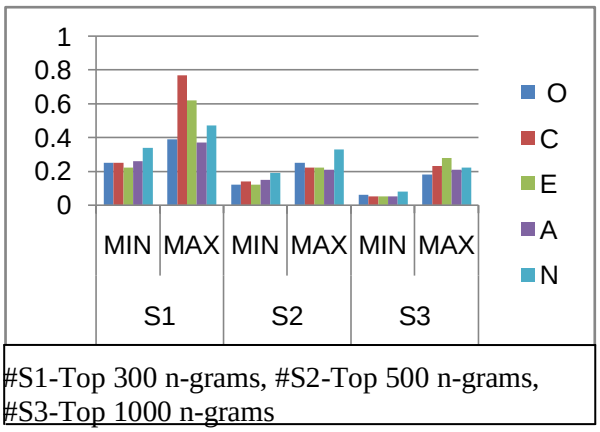
$$T_u^c = Total\ number\ of\ unique\ unigram\ of\ that\ class$$

$$C_w = Total\ occurence\ of\ that\ word$$

Fig 3: Equation for counting n-gram probability

The range of probability is identified by the lowest and the highest probability scores obtained for each personality class. For example, if the word *W1* occurs *X* times in a particular Big Five class say *Z* ,and the total number of unique unigram of Z class is Y, the occurrence probability of that word *W1* is *X/Y*. The occurrence probability is also calculated for both bigrams and trigrams. The probabilities of unigrams are shown in Figure 3.

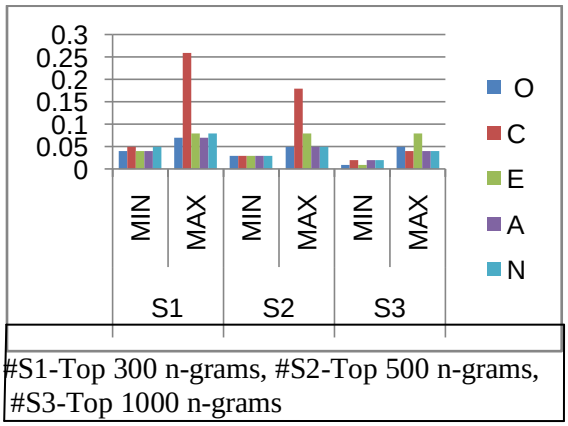
From our experiment, we observed that initially, we have started our experiment with top 300 n-grams and as we are interested in finding only those words that belong to only that specific class, so we apply two tier filtering. However, in order



#S1-Top 300 n-grams, #S2-Top 500 n-grams, #S3-Top 1000 n-grams

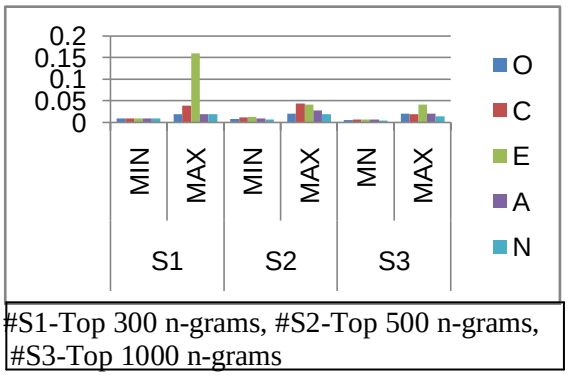
Fig 4: The occurrence probability graph of unigram

to follow this technique, we achieved very less number of words and phrases. Thus, we continue our experiments with top 500 and 1000 n-grams. While increasing the size, we observed that.



#S1-Top 300 n-grams, #S2-Top 500 n-grams, #S3-Top 1000 n-grams

Fig 5: The occurrence probability graph of bigram



#S1-Top 300 n-grams, #S2-Top 500 n-grams, #S3-Top 1000 n-grams

Fig 6: The occurrence probability graph of trigram

we have obtained more number of words and phrases But, we get some words and phrases whose individual frequency is very less and thus their occurrence probability is also very less in that class. Therefore, we can conclude that the words do not have any influence in classification.

4.3 Evaluation

Now, we want to test it against on another dataset. For testing, the dataset used is YouTube Dataset. A simple algorithm for testing is defined in Figure 7. We have adopted a strict evaluation scheme such that each of the test documents should belong to only one personality class. We do not get satisfactory result. Overall we get 35% accuracy.

5 LIWC based Classification Module

The advancements in the field of personality trait classification till could not answer the question that which features are most significant for personality

Step 1. *Split the text into sentences.*

Step 2. *Preprocess the dataset.*

Step 3. *Categorize each word of a sentence using Big Five personality lexicon*

Step 4. *A sentence is classified into one of the Big Five personality class based on the majority class of the words of that sentence*

Step 5. *A file is classified into one of the Big Five personality classes based on the majority class of the sentences of that file.*

Fig 7: Algorithm for testing Youtube dataset

classification. Thus, initially, we have started with some basic approach to build a lexicon for Big Five personality class. However, we could not achieve satisfactory results and thereafter we use LIWC, a widely used text analysis tool to improve our result.

5.1 LIWC

Linguistic Inquiry and Word Count, is a widely used text analysis tool to efficiently classify texts. We extracted 69 features of LIWC for each of the documents. Then, we tried to reduce the size of the feature set. The classifiers are then built on the re-

duced set of features and the performances are compared with respect to a complete set of features. This research aims to show that the usefulness of LIWC on personality identification and how different feature reduction techniques such as Information Gain, PCA can help in getting better result for classification.

5.2 Feature Extraction

LIWC was developed by Pennebaker et al., 2007. It is a text analysis tool that is employed to quantify features and allowed for text classification and prediction. LIWC is a dictionary that contains 80 categories. For each file, we consider each word and search through the dictionary. If the target word is found in the dictionary, the category count of that word is incremented. Though the dictionary contains 80 features, we initially started with our experiment on 69 features. We count the number of *anger, sad, pronoun, posemo* (positive emotion word), *negmo* (negative emotion word) etc. Based on the method, each file of essay dataset was fed into the LIWC. The output file contains 69 features and each feature has one of the Big Five personality traits as the classification label.

5.3 Classification

In order to validate the feature set, a number of experiments have been performed to evaluate how accurate they are in predicting Big Five personality traits. A 10-fold cross validation was performed on our feature set to assess the accuracy. We tested a number of popular classification algorithms like Support Vector Machine (libSvm), SMO, Multilayer Perceptron and Simple Logistic Regression.

5.4 Feature Selection

The feature set is reduced by selecting a subset of original features. The removed features are not used in classification anymore. One of the aims of feature selection methods is to determine a subset of features for which the accuracy is maximized.

5.4.1 Information Gain

As we want to determine which attribute in a given set of training feature vectors is most useful we use information gain. Information gain tells us how important a given attribute of the feature vector is thus helps in reducing the feature set size while keeping the accuracy same. One of the most important contributions of this research is to determine the most important features among the 69 LIWC features that can used for classifying the Big Five personality, keeping the accuracy same or making it better.

By considering the top 10 LIWC features of Information Gain, the obtained result was not satisfactory. Then, on increasing the size of the LIWC feature set with 15 more features, the result was not improved. Finally with a LIWC feature set of size 20, the result is nearly the same when compared to the result that is obtained with a LIWC feature set of size 69. Thus, in future, our aim will be to strengthen the feature set by extracting features from other lexicons like MRC, NRC and other optimization techniques like PCA.

5.4.2 Correlation Attribute Evaluation

After extracting features using LIWC, we wanted to reduce the feature set size and that's why we apply Information Gain. Now, we use another feature reduction technique, Correlation Attribute that evaluates the worth of an attribute by measuring the correlation between it and the class.

5.5 Result Analysis

Initially, the experiment has been performed on 69 features. We achieved better result on Libsvm on *radial basis function* kernel compared to Libsvm on *polynomial* kernel. Next, we tried to reduce our feature set. We test our result using two feature reduction techniques, one is Information Gain and another is Correlation Attribute evaluation. We test our results in two dataset, one is Essay dataset and another one is Youtube dataset. In essay dataset, we obtained very good result (accuracy of 78%) and in Youtube dataset we achieved 56% accuracy. In Table 1 and Table 2, we give the details of result.

5.5.1 Feature Level Analysis

In this experiment, we have observed the influence of different LIWC features on classification. We have done our experiment with different variations of features and tried to analysis the Precession (P),

Recall (R) and F-measure (F) to identify the importance of different features.
From LIWC, we started our experiment with 69 categories of words. Using Information Gain, when we ranked the attributes, we obtained top 10 features like **home, we, job, inhib (**inhibition**), excl (**exclusive**)** etc. which are very important importance of different features.

From LIWC, we started our experiment with 69 categories of words. Using Information Gain, when we ranked the attributes, we obtained top 10 features like **home, we, job, inhib (**inhibition**), excl (**exclusive**)** etc. which are very important categories for classification. Then, when we increase the size, the categories like **occup (**occupation**), leisure, anger** are added. Finally, when we considered top 20 features, we achieved the best classification result and some important features like **sad, negmo (negative emotions)** which were added further. Thus, we can say that among 69 features, these features have more importance than other features.

Using correlation attributes and when we rank the attribute under top 10 features, we get features like **smile, you, home, posfeel (**positive feeling**)** etc. Then, increasing size, we obtained features like **friends, time, school, eating** etc. Finally, while considering top 20 features, we achieved the best result on some features like **we, past, family, achieve, see** etc. as mentioned in Table 3 and Table 4.

6 Observation and Conclusions

A Personality Lexicon for Big Five Personality classes have been developed. The main objective is to find out some unique words that are mostly used by a particular type of personality. According to the design module, a lexicon with top 300, 500 and 1000 n-grams has been obtained. Our observation says when we continue our experiment with top 300 n-grams, the size of our lexicon is small and as we increase it to 500 n-grams and 1000 n-grams our lexicon size increases but it also contains words whose frequency in the text are very less.

On the other hand, in case of calculating occurrence probability of individual word belonging to a particular personality class, we observed some issues. When we take top 300 word, the occurrence probability is very high and as we take top 500 and 1000 n-grams, the occurrence probability decreases. As a result they do not help us much in classification. Thus, we can conclude that When we take top 300 n-grams, we get best result.

We developed our lexicon based on Essay dataset and tested it on YouTube dataset. As there is no topic related restriction on both dataset, the datasets contains diverse topics and that makes our job more difficult for personality identification and thus to develop a proper lexicon of a personality class becomes more difficult.

For development of lexicon, we already have discussed that we used two levels of filtering to eliminate all the words except a few which belongs to a particular class only. In order to maintain this process, we eliminate many words that may be important for us in classification. For example, the frequency of word **"Strange"** occurs in **Openness** class is **239** times and in **Extrovert** class is **20** times as because we are interested to find only those words that belong to **Openness** class. We eliminate the word **"Strange"** from the lexicon of Openness**.** As frequency of the words "strange" is so high in open class, so it may be an important unigram for the Openness class. So, for better classification result, we have to apply some threshold value which can be a future prospective of thiswork.

By using only n-gram approach we didn't get satisfactory result .Then we use LIWC for classification and we get very good result. Then we try to reduce the feature set by reducing the size of the feature set while keeping the accuracy same. We then use information gain optimization technique and reduce the size of the feature set from 69 to 20 while keeping the accuracy same.

Classifier	Accuracy (in %)							
	(Size = 69)		(Size = 10)		(Size = 15)		(Size = 20)	
	#IG	#CRA	#IG	#CRA	#IG	#CRA	#IG	#CRA
Libsvm(#1)	78.52	78.52	**78.18**	73.48	**78.52**	77.51	**78.52**	78.52
Libsvm(#2)	68.45	68.45	67.11	66.44	67.78	66.77	65.77	65.10
Multilayer Perceptron	63.75	63.75	33.55	35.23	42.61	41.27	47.31	56.71
Simple Logistic	41.94	41.94	30.20	31.87	28.18	32.88	30.20	32.88
SMO	38.92	38.92	25.50	30.53	26.84	31.20	29.86	35.23
Libsvm(#1):libsvm with on radial kernel. Libsvm(#2):libsvm with on polynomial kernel.#IG: Information Gain. #CRA: Correlation Attribute.								

Table 1: Result Analysis on different size feature set and on different classifier on *Essay* dataset

Classifier	Accuracy (in %)							
	(Size = 69)		(Size = 10)		(Size = 15)		(Size = 20)	
	#IG	#CRA	#IG	#CRA	#IG	#CRA	#IG	#CRA
Libsvm(#1)	56.60	56.60	56.60	56.60	56.60	56.60	56.60	56.60
Libsvm(#2)	45.28	45.28	45.28	47.16	37.73	30.18	41.50	39.62
Multilayer Perceptron	49.05	49.05	56.60	45.23	52.83	47.16	43.39	50.94
Simple Logistic	49.05	49.05	52.83	**62.26**	58.49	**62.26**	52.83	**60.37**
SMO	56.60	56.60	56.60	56.60	54.71	56.60	54.71	56.60
Libsvm(#1):libsvm with on radial kernel. Libsvm(#2):libsvm with on polynomial kernel.#IG: Information Gain. #CRA: Correlation Attribute.								

Table 2: Result Analysis on different size feature set and on different classifier on *Youtube* dataset

FEATURE	#NOF	#P	#R	#F
Eating, Home, we……home, job	10	0.86	0.78	0.79
We, Insight, occup……Other, Excl, Anger	15	0.88	0.78	0.80
See, prep, sad, motion…….anger, we, job, home	20	0.88	0.78	0.80

#NOF=number of file, **#P=**precision, **#R=**Recall, **#F=**F-measure.

Table 3: Feature selection using Information Gain and analysis with respect to precession, Recall and F-measure

FEATURE	#NOF	#P	#R	#F
Smile, you, home, sports…….senses.	10	0.82	0.73	0.74
Friends, time, school…….smile, leisure	15	0.87	0.77	0.79
we, past, family……..home ,achieve ,school	20	0.88	0.78	0.80

#NOF=number of file, **#P=**precision, **#R=**Recall, **#F=**F-measure.

Table 4: Feature selection using Correlation Attribute and analysis with respect to precession, Recall and F-measure

References

Alam Firoj, Evgeny A. Stepanov, and Giuseppe Riccardi. "Personality traits recognition on social network-facebook." *WCPR (ICWSM-13), Cambridge, MA, USA* (2013).

Burger, John D., John Henderson, George Kim, and Guido Zarrella. "Discriminating gender on Twitter." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1301-1309. Association for Computational Linguistics, 2011.

Biel, Joan-Isaac, Vagia Tsiminaki, John Dines, and Daniel Gatica-Perez. "Hi youtube!: Personality impressions and verbal content in social video." In *Proceedings of the 15th ACM on International conference on multimodal interaction*, pp. 119-126. ACM, 2013.

Biel, Joan-Isaac, and Daniel Gatica-Perez. "The youtube lens: Crowdsourced personality impressions and audiovisual analysis of vlogs." *IEEE Transactions on Multimedia* 15, no. 1 (2013): 41-55.

Baccianella, Stefano, Andrea Esuli, and Fabrizio Sebastiani. "SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining." In *LREC*, vol. 10, pp. 2200-2204. 2010.

Cambria, Erik, Catherine Havasi, and Amir Hussain. "SenticNet 2: A Semantic and Affective Resource for Opinion Mining and Sentiment Analysis." In *FLAIRS conference*, pp. 202-207. 2012.

Farnadi, Golnoosh, Shanu Sushmita, Geetha Sitaraman, Nhat Ton, Martine De Cock, and Sergio Davalos. "A multivariate regression approach to personality impression recognition of vloggers." In *Proceedings of the 2014 ACM Multi Media on Workshop on Computational Personality Recognition*, pp. 1-6. ACM, 2014.

Gupta, Umang, and Niladri Chatterjee. "Personality traits identification using rough sets based machine learning." In *Computational and Business Intelligence (ISCBI), 2013 International Symposium on*, pp. 182-185. IEEE, 2013.

Golbeck, Jennifer, Cristina Robles, and Karen Turner. "Predicting personality with social media." In *CHI'11 extended abstracts on human factors in computing systems*, pp. 253-262. ACM, 2011.

Golbeck, Jennifer, and Eric Norris. "Personality, Movie preferences, and recommendations." In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp.1414-1415. ACM, 2013.

Hu, Rong, and Pearl Pu. "Enhancing collaborative filtering systems with personality information." In *Proceedings of the fifth ACM conference on Recommender systems*, pp. 197-204. ACM, 2011.

Iacobelli, Francisco, and Aron Culotta. "Too neurotic, not too friendly: structured personality classification on textual data." In *Proc of Workshopon Computational Personality Recognition, AAAI Press, Melon Park, CA*, pp. 19-22. 2013.

Kelly, E. Lowell, and James J. Conley. "Personalityand compatibility: a prospective analysis of marital stability and marital satisfaction." *Journal of personality and social psychology* 52, no. 1 (1987): 27

Mairesse, François, Marilyn A. Walker, Matthias R. Mehl, and Roger K. Moore. "Using linguistic cues for the automatic recognition of personality in conversation and text." *Journal of artificial intelligence research* 30 (2007): 457-500..

.

Mohammad, Saif M., and Svetlana Kiritchenko. "Using nuances of emotion to identify personality.

" *Proceedings of ICWSM* (2013).

Max. Coltheart. 1981. . "the mrc psycholinguistic database."". *The Quarterly Journal of Experimental Psychology* ,, 33.

Oberlander, Jon, and Scott Nowson. "Whose thumb is it anyway?: classifying author personality from weblog text." In *Proceedings of the COLING/ACL on Main conference poster sessions*, pp. 627-634. Association for Computational Linguistics, 2006.

Pratama, Bayu Yudha, and Riyanarto Sarno. "Personality classification based on Twitter text using Naive Bayes, KNN and SVM." In *Data and Software Engineering (ICoDSE), 2015 International Conference on*, pp. 170-174. IEEE, 2015.

Pervaz, Ifrah, Iqra Ameer, Abdul Sittar, and Rao Muhammad Adeel Nawab. "Identification of Author Personality Traits using Stylistic Features: Notebook for PAN at CLEF 2015." In *CLEF (Working Notes)*. 2015.

Pennebaker, James W., and Laura A. King. "Linguistic styles: language use as an individual difference." *Journal of personality and social psychology* 77, no. 6 (1999): 1296.

Patra, Braja Gopal , Hiroya Takamura, Dipankar Das, Manabu Okumura and Sivaji Bandyopadhay.”Construction of Emotional Lexicon Using Potts Model.”In IJCNLP ,pp.674-679.2013.

Rangel F., Celli F., Rosso P., Potthast M., Stein B., Daelemans W. Overview of the 3rd Author Profiling Task at PAN 2015 . In: Cappellato L., Ferro N., Jones G., San Juan E. (Eds.) CLEF 2015 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings. CEUR-WS.org, vol. 1391.

Sidorov, Grigori, and Noé Alejandro Castro-Sánchez. "Automatic emotional personality description using linguistic data." *Research in computing science*20 (2006): 89-94.

Tausczik, Yla R., and James W. Pennebaker. "The psychological meaning of words: LIWC and computerized text analysis methods." *Journal of language and social psychology* 29, no. 1 (2010): 24-54.

Torii, Yoshimitsu, Dipankar Das, Sivaji Bandyopadhyay, and Manabu Okumura. "Developing japanese wordnet affect for analyzing emotions." In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pp. 80-86. Association for Computational Linguistics, 2011.

Tett, Robert P., Douglas N. Jackson, and Mitchell Rothstein."Personality measures as predictors of Job performance: a meta-analytic review." *Personnel psychology* 44, no. 4 (1991):703-742.

Wilson,Michael, “MRC psycholinguistic database: Machine usable dictionary,version 2.00” .Behavior Research Methods 20 .no. 1(1988): 6-10.

Transfer of Polarity Score for Sentiment Classification in Hindi

Vartika Rai **Sakshee Vijay** **Dipti Misra Sharma**
MT-NLP Lab,Kohli Center on Intelligent Systems
International Institute of Information Technology, Hyderabad -India
{vartika.rai,sakshee.vijay}@research.iiit.ac.in
dipti@iiit.ac.in

Abstract

Sentiment analysis in a resource scarce language is a tedious task. We propose a novel method for transfer learning from a target language to English. Our system doesn't rely on labeled data for the target language but instead links itself onto already existing and extensively labeled word-level lexical resource in English (ESWN[1]) and a semantic parser. Our proposed system transparently needs no target language sentiment corpus, and exploits complex linguistic structure of the target language for sentiment prediction. This cross lingual approach gives net accuracy as 83.6%, an improvement of 5.4% over the baseline system.

1 Introduction

In late 2000's, Hindi had least share in terms of online presence. English and European Languages had major share on web and social platforms.But after 2010, its presence has witnessed a sharp growth in web texts, social media platforms, online personal assertive tools, etc. There are over 200 million Hindi speakers in north India alone. With more and more people indulging themselves into using Hindi as their communication language, this huge amount of user generated corpus has created a strong need to exploit Sentiment Analysis of online web texts . Opinion Mining of these texts can open a big door to not only this language's and its speakers' properties but also the culture and practices of that language.

Sentiment Analysis is a natural language processing task that tries to identify nature of opinion in a piece of text. It can be with respect to a sentence, document or even aspects in sentences.
Key methods to extract/predict sentiment can be classified into three types.

- Using Machine Learning - Predicting data by applying supervised or semi supervised approach on features from the text .
- N-Gram Modeling/Bilingual Mappings : Using N-gram models along with training data for sentiment prediction.
- Using Subjective Lexicon : A Resource of words or group of words (phrases) with a polarity score assigned to each word. Score in this case points towards the properties bore by that word for categorization into positive, negative or neutral.

As Hindi is resource scarce language in terms of standard and labeled datasets, dependence on datasets which have low recall and coverage for classification tasks result in low precision and accuracy.
To solve this problem, a combination of above approaches have resulted in what we call Transfer Learning or Cross Lingual based approach,which is the task of predicting sentiments by testing in text of language L_target (in this case, Hindi) by using a classifier trained/labels on the corpus of another language L_source(English). This paper adopts the above approach of transfer learning between L_target and L_source to predict sentence level sentiment labels in Hindi text. We propose methods which are combination of

[1]http://sentiwordnet.isti.cnr.it/

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 373–382

above mentioned key methods. Method 1 uses Machine Learning techniques and classifiers such as Naive Bayes and SVM in predicting Sentence level score along with lexical resource to label data. Method 2 is a complete unsupervised approach which exploits highly accurate ESWN to label chunk level scores in sentence and hence calculate sentence level score using these chunk scores. Method 1 gives overall accuracy of 78.8 % and method 2 results in accuracy of 83.6 %.

2 Challenges

- **Weak Lexical Resources:** Sufficient resources like labeled data, Sentiment Tagged words, tools and annotated data for Hindi language are not available, and those which are available are not as good in coverage and accuracy standard as per its English counterpart. And Annotated corpora and tagger for Hindi language is not as good compared to English language, which makes the sentiment analysis task time consuming.

- **Free Word Order:** Word order plays important role in polarity detection. Hindi is a free word order language means there is no specific arrangement of words in Hindi language i.e. subject(S), object(O) and verb(V) comes in any order whereas English is fixed word order language i.e. subject-verb-object(SVO). Word order has a significant role in determining polarity of word and hence of sentences, documents which it is a building unit of.Even the slightest variations and changes in the word order affect the polarity label.

- **Multiple senses:** Words in Hindi language having same semantic meaning may occur in multiple contexts, making it tough to distinguish between senses and hence pick one of them.

- **Morphological Variations:** Hindi language is morphologically rich which means that lots of information is incorporated already in the words as compared to the English language.

- **Co-reference resolution:** Analysing multiple expressions that refer to the same thing. For example :

  – “गीता शाम को निकली और वह सब्ज़ियां खरीदने गयी” .
  – Transliteration : geeta shaam ko nikalee aur vah sabziyaan khareedane gayee
  – English : Geeta got out in the evening and she went to buy vegetables. “वह” also refers to गीता. This analysis is important while performing fine grained level sentiment analysis.

3 Literature Survey

A lot of work has been done until now in the field of sentiment analysis for Hindi language with purpose to classify text and create lexical resource. Existing multilingual and cross lingual sentiment analysis approaches involve extension of existing resources through translation, synset, concept linking to bridge the gap. Recent methods based on learning common vector spaces for multiple languages have also shown promise in some topics.
In terms of creating lexical resources and extensions, most notable contributions are from Amitava Das and Bandopadhya [1],in which they developed sentiwordnet for Bengali language by Word level lexical-transfer technique on English SentiWordNet using an English-Bengali Dictionary. They also devised four approaches to predict polarity of a word in [2]

A Fallback strategy was proposed by Joshi et al. in [3] for Hindi language to create lexical resource Hindi SentiWordNet (HSWN) based on its English format. H-SWN (Hindi-SentiWordNet) by using two lexical resources (English SentiWordNet and English-Hindi WordNet Linking) with using methods namely: In-language Sentiment Analysis, Machine Translation and Resource Based Sentiment Analysis. Bakliwal et al.[4] created resource using a graph based method .They depicted how the synonym and antonym relations can be used to generate the subjectivity lexicon by using the simple graph traversal approach with 79% accuracy on classification of reviews.

A Graph based method was proposed by Piyush Arora et al. [5] to build a subjective lexicon for Hindi language, using WordNet as a graph traversal resource. Small seed list of opinion words was initially built and by using WordNet and synonyms and antonyms of the opinion words were determined and added to the seed list. An efficient approach was developed by Namita mittal et al. [6] based on negation and discourse relation to identifying the sentiments from Hindi content by improving Hindi SentiWordNet (HSWN) by adding more entries. They also created the rules for handling negation and discourse and 80% accuracy was achieved by their proposed algorithm for classification of reviews.

Various alterations to features in training set with supervised approaches have been used in [7] [8] [9] [10] [11] [12] [13]

A simple technique to perform sentiment classification based on an unsupervised linguistic approach using SentiWordNet to calculate overall sentiment score of each sentence is expressed in [14].

In terms of approaches which involves cross lingual methodology,which means training in source language L_source and testing on target language L_target, following notable works have been published. Using an english dataset, two Hindi language training datasets are produced with different features by [15]. Balamurali (2012) used WordNet senses as features for supervised sentiment classification. They use the linked WordNets of two languages to connect the languages. [16]. Deep learning framework is used in [17] to learn feature representations for cross lingual approach.

But all of these approaches require at least some amount of labeled data and complete inhouse resources with training data, heuristics in that language and in case of cross lingual approach, involves dependency on resource of source language.

4 Experimental Setup

We conducted two experiments, one with dependency on Hindi lexical resource and supervised in nature and another unsupervised in nature with its dependence on lexical resource in English. For supervised approach, classifiers such as Naive Bayes and SVM are used, and for unsupervised approach, Google Translate is used translate chunks into L_target English and then we interlink chunk level sentiments to L_source for further processing.

4.1 Datasets

We have used data from following resources

4.1.1 Data Used

- Hindi News Sentences [18]
- English SentiWordnet(ESWN) as a lexical resource reference. [19]
  Entries in this resource is modified as per need of our task. Hence, for every word, it matches the POS tag, most common or most frequent used sense and then returns score as a tuple, in which first entry is the positive score of word and second entry is the negative score.Since we already know that second entry is the negative score, we do not necessarily put '-' (minus sign) infront of it, to indicate its negative nature.

  Examples:

  - ESWN_Score(good) : (0.75,0.0)
    It contains more than 8 senses, but it returns most commonly used sense.
  - ESWN_Score(evil) :
    (0.375,0.5) if it occurs as Noun and
    (0.0,0.875) if it occurs as adjective.
- Data extracted from websites such as
  www.patrika.com/gadgets/
  www.amarujala.com/
  aajtak.intoday.in/

4.1.2 Resource Contribution

2000 sentences with political domain as its background have been selected from above sources and have been manually labeled into three classes, Positive (P), Negative(N),& Neutral/Statement (S) on the basis of annotation guidelines.

Example:

लोगों को यह एक अच्छा व्यवसाय नजर आने लगा है।	**P**
भारत के लिए यह एक बुरा दिन साबित हुआ	**N**
नोटिस पर सुनवाई सोमवार को होगी	**S**

After one round of labeling sentence as positive,negative or neutral, the data was

distributed into couple of more annotators who also labeled the data according to their understanding. Hence, each sentence was labeled by 3 annotators. The sentence with at least 2 similar label out of 3 were considered and one more round of annotations were conducted for them. At the end, the label having >75% inter annotator agreement were incorporated. This data also acts as our evaluation model on experiments mentioned in this paper.

4.2 Supervised approach using Hindi lexicon

This experiments uses HSWN to label sentence level polarities. After receiving labeled sentence level polarities, classifier is run on this data to predict unseen sentences into one of the two classes, positive and negative.

4.2.1 Preprocessing

The first task is to run several iterations of processing on data, in which each given sentence is checked for noise and entries other than Hindi words.Spelling mistakes are corrected so that parser produces as accurate parse trees as possible.
In example mentioned below , the original word in corpus with its translated English equivalent is mentioned in bracket, and then , the same word after spelling correction and its correct English form is mentioned.
1. अभीन्न (Abinn) -> अभिन्न (Integral)
2. हॉलमार्क (Halmark) -> हॉलमार्क (Hallmark)
The final task is appending the missing end marker of sentence "|".

4.2.2 Feature Vector from Parser

Each hindi sentence is run through Shallow Parser [2] which produces a output which contains complete description of the word, its POS tag, its root form, morphological analysis and representation in WX notation. At a bigger level, chunks are also assigned heads and they have these properties too. Hence, whole sentence is now rich with linguistic features of all its words.
An example of feature set for a single word:
Hindi Word : असुविधा (English counterpart : Inconvenience)

[2] http://ltrc.iiit.ac.in/analyzer/hindi/

असुविधा NN <fs af=' असुविधा ,n,f,sg,3,d,0,0' name='asuviXA'>

4.2.3 Enhancing Feature Set

For each word in our parsed data, we incorporate not only its linguistic properties but also its polarity. We use lexical resource HSWN(Hindi SentiwordNet) [3] to retrieve polarities of words.

Algorithm 1

For Each word in Sentence
Search The word in HSWN :
if Present **then**
append that particular polarity as a feature into existing feature set
else
locate the English translated version in ESWN and append that polarity.
end if

While translating, sense in preserved by taking into account the POS tag. And in case of multiple senses present in the lexical corpus, we take into account the most commonly and frequent used sense. Hence, now we have a feature set which has both linguistic and polar properties.
Example :
अच्छा JJ <fs af='अच्छा,adj,m,sg,,d,,' name='अच्छा' posn='110',score = '0.75,0.0'>

4.2.4 Preparing Training Data

We convert our feature set into a metric of :
1) TF features
2) TF-IDF features.
The weight of a term that occurs in a documents is simply proportional to the term frequency, while a term's inverse document frequency (idf) is inverse function of the number of documents in which it occurs.And,

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \quad (1)$$

Python module Scikit Feature Extraction Module was used to perform this. Feature size obtained through this is roughly 20,000 which is pretty large to process with good results. Hence, we apply dimensionality reduction techniques such as Principal Component Analysis(PCA) [20] to reduce the feature size by 1/4th and the feature set now has a size of 5000.

4.2.5 Assigning score to sentence

The net score of sentence is weighted summation of polarity scores of all of its words. These weights are designed through special heuristics which is mentioned below.

A. Negative Dominance
It states that given a sentence, if the sum of negative polarity of words is greater than 50% of sum of positive polarity, we classify the sentence as negative.

B. Chunk Rule
Given a chunk (format in which the sentences occur in dataset), if a chunk contains a NEG tag, it reverses the polarity of all the words and hence the sentence till then.

C. Inflected Case
The equations Given a word with inflected form, it is not necessary to have its polarity similar to that of root word and has different polarity assigned to it in HSWN. But in case, the inflected word in unavailable, we tend to derive its polarity from its root word, root word being detected from the feature vector produced by shallow parser (the second word depicting the lemma(root) of the word).

4.2.6 Classifiers

Naive Bayes and SVM are run onto this to classify sentence as Positive or Negative. Accuracy is measured through manually labeled corpus mentioned in contribution.

Naive Bayes

A Naive Bayes Classifier is based on Bayes' theorem and is particularly used when the input dimensions are high. Naïve Bayes classification is a text classification approach that assigns the class c to a given document d as :

$$c* = argmax_c P(c/d) \quad (2)$$

Where P(c|d) is the probability of instance d being in class c [21].

SVM

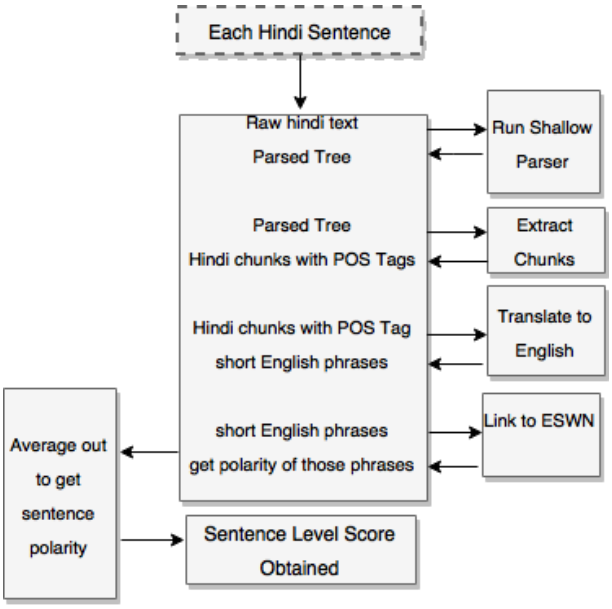
Support Vector Machine Classifier constructs N-dimensional hyper-plane represented by vector $\vec{w}$ which separates data into two categories.SVM takes the input data and for each input it predicts the class. SVM can be seen as a constrained optimization problem, in which class

$$c_j 1, -1 \quad (3)$$

corresponds to either positive or negative class that belongs to document d_j , the solution can be written as :
$\vec{w} = \sum_j \alpha_j c_{jd_j}, \alpha_j >= 0$
Where $\vec{w}$ is a vector, c_j is a class and d_j is a document [22].

4.3 Unsupervised Approach with Transfer Learning from English to Hindi

This approach works on the basis that each sentiment bearing word polarizes the words near it and hence the polarity around that word is similar to that of word. So, for each sentence, we extract chunk based polarity, with assumption that each polarity bearing word/words assign polarity to the whole chunk, and instead of using lexical resource in Hindi, we depend on already existing and quite accurate lexical resource in English, English SentiWordNet, to extract polarity scores for those Hindi to English translated chunks and then we transfer each chunk level score back to its original Hindi chunk, hence labeling every Hindi chunk with polarity score one by one. This approach rules out the dependence on Hindi labeled data and other resources which isn't that rich compared to ESWN.



Parsed Tree Example :

1	((	NP <fs af=शाम,n,f,sg,3,d,0में,0 head="शाम">
1.1	शाम	NN <fs af=शाम,n,f,sg,3,d,0,0 name="शाम">
1.2	को	PSP <fs af=को,psp,,,,,'>
	))	
2	((	NP <fs af=मौसम,n,m,sg,3,d,0,0' head="मौसम">
2.1	मौसम	NN <fs af=मौसम,n,m,sg,3,d,0,0' name="मौसम">
	))	
3	((	JJP <fs af=अच्छा,adj,m,sg,,d,, head="अच्छा">
3.1	बहुत	INTF <fs af=बहुत,n,m,sg,3,d,0,0' poslcat="NM">
3.2	अच्छा	JJ <fs af=अच्छा,adj,m,sg,,d,,' name="अच्छा">
	))	
4	((	VM <fs af=हो,v,any,any,any,,0,0' name="हो">
4.1	हो	VM <fs af=हो,v,any,any,any,,0,0' name="हो">
4.2	गया	VAUX <fs af=जा,v,m,sg,any,,या१,yA1' poslcat="NM">
	))	

Table 1: Shallow Parser Output

4.3.1 Sentence Parsing:

Given a sentence , Shallow parser is run on it, which gives complete analysis of a sentence in terms of Part of Speech , Morphology, Chunking etc. By using these properties, we will be able to predict overall sentiment score.

Algorithm 2

for Each Sentence S: **do**
  S_parsed = Shallow Parser(S)
end for

For example, given the sentence :
Hindi : शाम को मौसम बहुत अच्छा हो गया.
Transliteration : Shaam Ko Mausam Bohot Achha Ho Gaya
English : Weather got really good in the evening.
Shallow parser output is shown in Table 1(above)
As seen, the whole sentence can be represented as group of various chunks, with chunk heads. We extract these chunks along with their POS tags, and proceed to chunk processing step.

4.3.2 Chunk Processing Step:

In this step, we have chunks of sentences with their POS tags. The algorithm for this is :

Algorithm 3

for each sentence S **do**:
  for each chunk C in all chunks of S:
 do
  Translate(Chunk_hindi)-(Chunk_english)
  end for
end for

In this step, each Hindi chunk of a sentence in translated to english,& since we are translating chunk with max of 5-6 words per chunk,expected translation error is pretty low as compared to translation of complete sentence, which will help us to effectively map sentiments,without using any hindi resource and with assumption that sentiment remains constant across these chunks.

4.3.3 Finding Sentiment:

Now, Each sentence S is a group of translated english chunks(E_Chunks). For each english chunk, we find its sentiment according to following algorithm:

Algorithm 4

for for each english chunk C in E_Chunks: **do**
  for for each word w in chunk C: **do**
    if if word in ESWN and its POS tags match **then**
      assign score to the word.
    else if Word is present but not in root form **then**
      convert word to its root and go to step 1
    else

assign score as 0,0.
end if
end for
end for

This algorithm assigns each word with a specific polarity taking into account its context as well. POS tags of words are used to distinguish between word senses, For example:
In the given phrase, ' he has the will to live', the word 'will' is having NN as its POS tag, while in the phrase, 'I will go there tomorrow', the word 'will' has VB tag.

So, it will have different score with respect to its manner/sense of occurrence in the sentence. Therefore, the POS tags need to match with the one in lexical corpus to match the correct sense and therefore attach correct score to the word. For words with multiple senses and hence different scores, the most commonly used sense is used for reference.
After marking each word with polarity score, we can have two approaches to assign score to the chunk:

Algorithm 5

if if a chunk contains more than one polarity bearing word: **then**
chunk_score = max(score of all polarity bearing words)
else
chunk_score = score of polarity or opinion bearing word
end if

This step assigns a polarity score to the current translated English chunk. Now, we transfer this score back to its original untranslated chunk, and after retrieving polarity scores all chunks, we calculate sentence level polarity by averaging out the chunks' score with total number of chunks.
Here, total number of chunks are those which actually contain any amount of polarity score and are not completely neutral.
So, an important point to note in this step is that not every chunk contributes to the overall polarity score of a sentence, while some chunks might have net polarity as (p:0,n:0), other might not have any polarity score due to their semantic and syntactic space, and their strict objective nature.

Example 1(positive label)

- Sentence: व्यापार में बेहतर काम उपभोक्ताओं के लिए लाभप्रद होता है ।
- Transliterated : vyaapaar mein behatar kaam upabhoktaon ke lie laabhaprad hota hai
- English : Better work in business is profitable for consumers.
- Chunked: (व्यापार में)_NP (बेहतर काम)_NP (उपभोक्ताओं के लिए)_NP (लाभप्रद)_JJP (होता है)_VGF
- English Chunks: (in buisness)_(better work)_(for the consumers)_(profitable)_(happens).

English Chunks	Polarity(pos,neg)
in business	(0.0,0.0)
better work	(0.875,0.0)
for the consumers	(0.0,0.0)
profitable	(0.25,0.0)
happens	(0.0,0.0)

Determining Average Polarity

Net average Polarity = (average positive polarity,average negative polarity)

- Average Positive Polarity : sum of all positive scores in chunks/ number of chunks having score >0
- Average Negative Polarity : sum of all negative scores in chunks/ number of chunks having score >0

Following the mentioned steps,in this case
Average polarity : (0.56,0.0)
Since |positive polarity| > |negative polarity|
Label Generated : Positive

Example 2(negative label)

- Sentence: सही एडिटिंग न होने के कारण दूसरे हिस्से में यह फिल्म कमजोर हो जाती है।
- Transliterated : sahee editing na hone ke kaaran doosare hisse mein yah philm kamajor ho jaatee hai.
- English : Due to lack of proper editing, this film becomes weak in the second part.

- Chunked: (सही एडिटिंग)_NP (न होने के कारण)_NP (दूसरे हिस्से में)_NP (यह फिल्म)_NP (कमजोर)_JJP (हो जाती है)_VGF
- English Chunks :
  (correct editing)_(Reasons for not being)_(In second part)_(this film)_(weak becomes)

English Chunks	Polarity(pos,neg)
correct editing	(0.625,0.0)
Reasons for not being	(0.0,0.675)
In second part	(0.0,0.0)
this film	(0.0,0.0)
weak becomes	(0.125,0.5)

Average polarity : (0.25, 0.4)
Since |negative polarity| > |positive polarity|
Label Generated : Negative

4.4 Improvisation over Previous Experiment

A different scenario is observed when a chunk contains Negation tag. In most of the cases, It is seen that it negates the chunk/word just previous to it. Therefore, presence if 'NEG' tag can alter the chunk level and hence sentence level polarity calculated through the previous experiment.So, we incorporate this factor too, while predicting sentiments.

- If a current chunk has 'NEG' tag:
  - *It nulls the polarity of previous chunk if it is positive*, and
  - *strengthens/adds up to the previous chunk score if its already negative.*

For example:

- Sentence: फिल्म की कहानी अच्छी नहीं है.
- Chunked: (फिल्म की)_NP (कहानी)_NP (अच्छी)_JJP (नहीं है)_VGF.
- English chunks:
  (film's)_(story)_(good)_(is not)

English Chunks	Polarity(pos,neg)
film's	(0.0,0.0)
story	(0.0,0.0)
good	(0.875,0.0)
is not	(0.0,0.625)

- Sentence polarity without negation handling: (0.44, 0.31)
  - Which is positive (wrong label)
- Sentence polarity after negation handling: (0.0, 0.675)
  - Which is negative (correct label)

4.5 Results

Results depict that with supervised approach, best case accuracy is 48.8% in case of Naive Bayes and 78.2 % in case of using SVM as our classifier,which is our baseline. In unsupervised transfer learning based approach, the accuracy is 83.6% which indicates the importance of lexical coverage and wideness if the experimental approach is corpus based.

Naive Bayes	**Result**
TF	40.8%
TF with heuristics	42.6 %
TF-IDF	44.6 %
TF-IDF with heuristics	48.8 %

Table 2: Naive Bayes

SVM	**Result**
TF	62.7%
TF with heuristics	64.4 %
TF-IDF	74.6 %
TF-IDF with heuristics	**78.2%**

Table 3: SVM Classifier

Experiment	**Result**
transfer learning sentence level	82.2 %
Transfer learning with negation handling	**83.6%**

Table 4: Transfer Learning

5 Conclusion

The experiments state one thing very clearly, that the performance of system is in accordance with the lexical resource at disposal, if any. When we used Hindi lexicon, the performance was not as good even though it was a supervised learning approach. The problem lies with the fact that Hindi SentiWordnet is limited in its coverage area and is not as extensive and rich in word level sentiment coverage

as its English counterpart. For example,Basic word such as 'नहीं ' isn't present in the list. So, most of the sentiment bearing words couldn't get sentiment labels, and the translation approach used to enhance the coverage depends on sense present in ESWN and acquired by parsed output. Although translator is not up to the mark for all time, as the sentence length shrink to 4-5 words, it performs decent enough to capture the underlying sentiment in that chunk.

Naive Bayes and SVM performance were hence, not very effective. When switched to lexical resource in L_target English, and unsupervised approach, the accuracy is increased because sentiment across language remains as preserved as possible because of minimal translation error and better coverage.

This approach is also important because the complete experiment depends on translation and ESWN and the problem of low coverage of lexical resource and no good training data in resource scarce language doesn't comes to picture.

One important thing using translator is the error while translating chunks having coreference to other part, or when the sentence structure is of a very casual conversation. For Example :

- अपने भविष्य की कोई खबर नहीं है उसे
- Transliterated : bhavishy kee koee khabar nahin hai use
- Translation Output : There is no news of his future
- Correct Translation : He has no idea of his Future.

This makes it difficult in capturing the essence of sentence and it becomes more of a general statement than a concern and hence looses the sentiment tag. While these errors were less in number as chunks were rarely greater than 3-4 words, its important to take these semantic points to get better idea of what's in the sentiment property of every sentence.

6 Future Work

Future work involves extension of our contributed dataset to aspect level and increasing its size to make it more useful and effective for purpose of Sentiment Analysis in various domains.

In second approach, incorporating more semantic and sense information while translating and taking into account the contribution of nearby chunks in determining a particular chunk polarity can increase the accuracy. The relation between chunks can help semantic properties intact. Also, a sophisticated mathematical model can be developed to figure out sentence level polarity instead of averaging the chunk scores.

References

[1] Amitava Das and Sivaji Bandyopadhyay. Sentiwordnet for bangla. *Knowledge Sharing Event-4: Task*, 2, 2010.

[2] Amitava Das and Sivaji Bandyopadhyay. Sentiwordnet for indian languages. *Asian Federation for Natural Language Processing, China*, pages 56–63, 2010.

[3] Aditya Joshi, AR Balamurali, and Pushpak Bhattacharyya. A fall-back strategy for sentiment analysis in hindi: a case study. *Proceedings of the 8th ICON*, 2010.

[4] Akshat Bakliwal, Piyush Arora, and Vasudeva Varma. Hindi subjective lexicon: A lexical resource for hindi polarity classification. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC)*, 2012.

[5] Piyush Arora, Akshat Bakliwal, and Vasudeva Varma. Hindi subjective lexicon generation using wordnet graph traversal. *International Journal of Computational Linguistics and Applications*, 3(1):25–39, 2012.

[6] Namita Mittal, Basant Agarwal, Garvit Chouhan, Nitin Bania, and Prateek Pareek. Sentiment analysis of hindi review based on negation and discourse relation. In *proceedings of International Joint Conference on Natural Language Processing*, pages 45–50, 2013.

[7] Amandeep Kaur and Vishal Gupta. A survey on sentiment analysis and opinion mining techniques. *Journal of Emerging Technologies in Web Intelligence*, 5(4):367–371, 2013.

[8] Braja Gopal Patra, Dipankar Das, Amitava Das, and Rajendra Prasath. Shared task on sentiment analysis in indian languages (sail) tweets-an overview. In *International Conference on Mining Intelligence and Knowledge Exploration*, pages 650–655. Springer, 2015.

[9] Subhabrata Mukherjee, Pushpak Bhattacharyya, et al. Sentiment analysis in twitter with lightweight discourse analysis. In *COLING*, pages 1847–1864, 2012.

[10] Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

[11] Akshat Bakliwal, Piyush Arora, Ankit Patil, and V Verma. Towards enhanced opinion classification using nlp techniques. In *Proceedings of the 5th international joint conference on natural language processing (IJCNLP). Chiang Mai, Thailand*, pages 101–107. Citeseer, 2011.

[12] Richa Sharma, Shweta Nigam, and Rekha Jain. Polarity detection movie reviews in hindi language. *arXiv preprint arXiv:1409.3942*, 2014.

[13] Shriya Se, R Vinayakumar, M Anand Kumar, and KP Soman. Predicting the sentimental reviews in tamil movie using machine learning algorithms. *Indian Journal of Science and Technology*, 9(45), 2016.

[14] Monalisa Ghosh and Animesh Kar. Unsupervised linguistic approach for sentiment classification from online reviews using sentiwordnet 3.0. *Int J Eng Res Technol*, 2(9), 2013.

[15] Poonam R Gohad and Archana S Vaidya. Hindi opinion mining for opinion target extraction. *International Journal of Engineering Science*, 6733, 2016.

[16] AR Balamurali. Cross-lingual sentiment analysis for indian languages using linked wordnets. 2012.

[17] Sarthak Jain and Shashank Batra. Cross lingual sentiment analysis using modified brae. In *EMNLP*, pages 159–168, 2015.

[18] Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *LREC*, pages 759–765, 2012.

[19] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204, 2010.

[20] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[21] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2):103–130, 1997.

[22] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Scalable Bio-Molecular Event Extraction System towards Knowledge Acquisition

Pattabhi RK Rao
AU-KBC Research Centre
MIT Campus of Anna University, Chennai, India
pattabhi@au-kbc.org

Sindhuja Gopalan
AU-KBC Research Centre
MIT Campus of Anna University, Chennai, India
sindhujagopalan@au-kbc.org

Sobha Lalitha Devi
AU-KBC Research Centre
MIT Campus of Anna University, Chennai, India
sobha@au-kbc.org

Abstract

This paper presents a robust system for the automatic extraction of bio-molecular events from scientific texts. Event extraction provides information in the understanding of physiological and pathogenesis mechanisms. Event extraction from biomedical literature has a broad range of applications, such as knowledge base creation, knowledge discovery. Automatic event extraction is a challenging task due to ambiguity and diversity of natural language and linguistic phenomena, such as negations, anaphora and co-referencing leading to incorrect interpretation. In this work a machine learning based approach has been used for the event extraction. The methodology framework proposed in this work is derived from the perspective of natural language processing. The system includes a robust anaphora and coreference resolution module, developed as part of this work. An overall F-score of 54.25% is obtained, which is an improvement of 4% in comparison with the state of the art systems.

1 Introduction

Tremendous growth in the field of biomedical science has resulted in large amount of clinical and biomedical medical data. Primarily, the biomedical research largely focused on genome data analysis. Over the years, the application of new technologies to health care has resulted in voluminous data that includes structured and unstructured clinical notes, patient data, imaging data, etc. This growth also resulted in accumulation of large number of biomedical texts, i.e. medical literatures. It is important to extract useful information from these data to benefit the researchers for further findings. This requires the application of data driven approaches. Data mining involves the analysis and extraction of interesting patterns from large amount of data. In recent times the researchers are spending much effort on data mining for bioinformatics. The previous applications of data mining and machine learning (ML) to bioinformatics were on genetic data sets and phenotype data. Now it has been extended to text documents like clinical and biomedical data.

In the early days, the goal of natural language processing in biomedical domain was to populate the databases with biological information. This can be done manually, but requires lots of effort and is time consuming. Hence recognizing the named entities (NEs) using computational techniques could help in automatically populating the database with biological information. The extraction of the information like event or relation between biomedical entities will help the research community to compare the applicability of their works with others. Finding related literatures studying same biomedical entities is a crucial and challenging task. For example, there are lots of research publications related to "BRCA" gene. Unifying all studies about this gene helps the researchers to work on cancer therapy. The first step for accomplishing this task is extracting the biomedical named entities from literature and finding the events and relations between them.

Therefore, mining the literature and extracting the event between biomedical entities have lots of applications in bioinformatics. Event extraction from scientific texts in biomedical domain such as PubMed abstracts has attracted a lot of interest in the last decade, especially for those events involving proteins and other biomolecules. In the biomedical domain, an event refers to the change of state of one or more biomedical entities, such as proteins, cells, and chemicals. In the task of event extraction we

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 383–391

need to identify the types of the events and their arguments. Event arguments include event participants, which may be entities (e.g., proteins) or other events. This structured definition of events is associated with an ontology that defines the types of events and entities, semantic roles, and also any other attributes that may be assigned to an event. Examples of ontologies for describing bio-molecular events include the Genia Event Ontology. Consider the below Figure 1

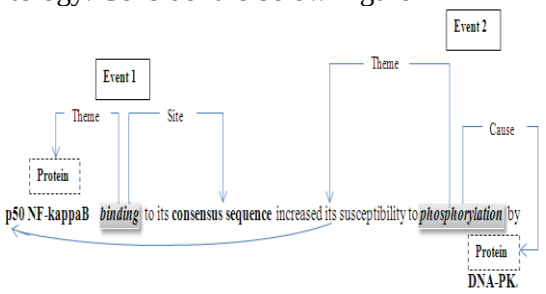


Figure 1: An Example for Event Occurrence in Biomedical text

The above Figure 1 shows two events, binding and phosphorylation. The first event belongs to event type binding, where the event has two arguments protein "p50 NF-kappaB", which is the theme and the second argument is the site "consensus sequence". The second event belongs to the event type phosphorylation and the arguments of this event is pronoun "its" which refers to the protein "p50 NF-kappaB", the theme of first event and protein DNA-PK, cause of second event. This example demonstrates the need for anaphora resolution in event identification.

By identifying the events we can extract information like gene-protein interactions, gene-chemical interactions and gene functions, etc. The BioNLP 2013 shared task on Genia Event extraction, has brought in more research groups to work in this area and has increased the research activity. Most of the systems which have participated in the BioNLP-ST 2013 Genia event (GE) extraction (Nédellec et al., 2013) have used the support vector machine (SVM) based pipeline. Two of the systems had used rule based approach. And another two had used hybrid approach where both rules and SVM have been used (Nédellec et al., 2013). In terms of use of pre-processing tools all the systems have used one of the deep parser tools such as McClosky-Charniak-Johnson Parser, Stanford Parser for syntactic processing. And some of the participating systems have also used external independent resource such as UniProt (Bairoch et al., 2005), IntAct (Kerrien et al., 2012), and CRAFT (Verspoor et al., 2012). Below we explain top two successful systems which have participated in the BioNLP-ST GE task. Both these systems have obtained an overall F-score of 50.97% and 50.74%.

Hakala et al., (2013) uses EVEX tool to extract the events. EVEX is a text mining resource built on top of events extracted from all PubMed abstracts and PubMed Central Open-Access full-text documents (Landeghem et al., 2011). Evex is built on top of "Banner" NER tool and "TEES" extraction tool. It uses SVM to re-rank the output of the EVEX resource output, sets a threshold score, below which the events are removed. The threshold score is obtained using a linear SVM regressor on each sentence. The results for event types "binding" and "regulation" are found to be lower and especially in "Methods" and "Captions" section of the documents.

Our work contributes to the application of data mining approach to biomedical data. This paper describes an event extraction system developed using ML approach and rich feature set including linguistic and biological domain motivated features. It has been observed from the participating systems in the BioNLP-ST 2013 that most of the systems have not used coreference resolution. Though the data had anaphora and coreference annotation, the systems had not exploited the annotations. In the present work we make use of the coreference annotations provided in the data. The use of coreference resolution has mainly improved the extraction of event type "binding".

The main contributions of this work are as follows:

1. We have developed a robust, scalable event identification system, which can be used for any of the biomedical domain documents. The developed system architecture is robust and portable for any biomedical text. The results obtained on the test data of the BioNLP ST 2013 GE task shows significant results comparable to the state-of the art.
2. We have developed a biomedical domain anaphora coreference resolution module for resolving the protein coreference relations. A general domain, robust anaphora coreference resolution module has been used and adapted (or customized) with the use of biomedical coreference annotations.
3. We have used open source tools such as "Genia tagger", CRF++ (Taku, 2005) for the development of the syntactic and semantic pre-processing. Thus this work is

easily implementable by other researchers.

In the following section we describe the corpora, features and the method used to develop the system. The results are discussed in Section 3. The paper ends with the conclusion and future works.

2 Method

The various approaches to event extraction task are rule based, dictionary based, ML and Hybrid approaches. This paper proposes a robust, scalable BioEventTag system developed by using graph-based ML technique Conditional Random Fields (CRFs) (Lafferty et al., 2001). This system is developed for the extraction of biological events. We have used CRF++ tool, an open source implementation of the CRFs algorithm. In this section, we present the experiments performed to extract the events from biomedical texts. First, the input text is syntactically preprocessed using Genia tagger. The pre-processing includes sentence splitting, tokenization, PoS tagging and chunking. Then in the next step semantic pre-processing is performed where the biomedical named entities (BNEs) are identified and anaphors in the document are resolved. For identifying the NEs we have used the biomedical named entity recognition (BioNER) system developed by (Gopalan et al., 2016). We developed an anaphora resolution system to resolve the anaphors. Finally we developed an event extraction system which contains two modules. First module identifies the event trigger and the second module extracts the event from the text. The system architecture is shown in Figure 2.

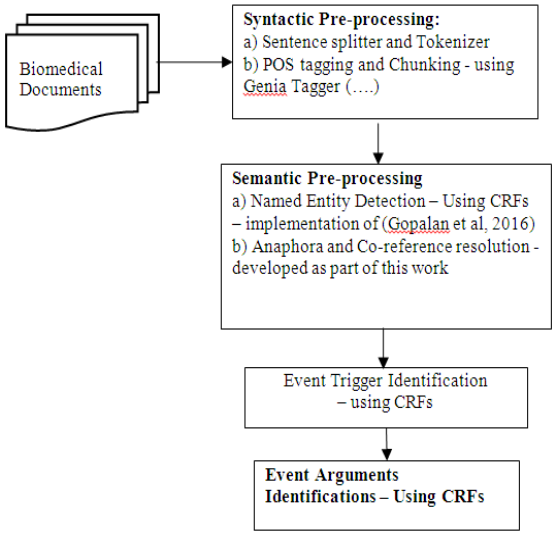


Figure 2: System Architecture

2.1 Corpora Collection and Analysis

We developed our system using a widely accepted dataset BioNLP-ST 2013 (Genia Event Task data). BioNLP-ST 2013 GE data was developed to evaluate the applicability of event extraction systems. The collection consists of 1210 titles and abstracts and 34 full papers from the Open Access subset of PubMed Central (Nédellec et al., 2013). Table 1 shows the corpus statistics. It is evident from the corpus statistics displayed in Table 1 that the majority of the events (65.86%) are "Regulation and Binding" event types. Thus handling of these two event types properly is very important to improve the system efficiency and performance. In these two types of events anaphora coreference resolution plays a significant role.

S.No	Description	Statistics
1	Number of Abstracts + full Papers	1210 + 34
2	Number of Words	2,63,133
3	Number of Proteins	16,427
4	Total Number of Events	9,364
5	Number of Anaphora and Coreference relations to Proteins	535
6	Number of Regulation and Binding Event types	6,168

Table 1: BioNLP Genia Event (GE) Shared Task Corpus Statistics

2.2 Feature Extraction

CRF++ is a general purpose tool and hence the feature template needs to be specified in advance. This file describes the feature used for training and testing. When the feature template is given, CRF++ automatically generates a set of feature function. The challenge in developing an event extraction system using ML techniques lies in designating the striking features and designing of feature template. We have used window size of 5 for this work. We describe in detail the features used in developing our system.

Lexical features and **Syntactic features** such as word, Parts of Speech (PoS) and chunk are used. PoS help in disambiguating the sense of the word in a sentence. PoS is an important feature for extracting the events as most of the arguments of an event are proper noun and event trigger belongs to noun and verb category. Hence PoS is a key feature for event extraction task. Most of the event trigger and arguments are descriptive i.e., they occur as a phrase. Hence

chunk tag will help in argument and event trigger extraction.

Morphological Patterns: Prefix/suffix is used as one of the features in our work. For example, an event trigger like "phosphorylation", has suffix '-ation', which means action or process. Prefix/suffix of a token helps to boost the performance of the system.

Biomedical Named Entities: BNEs are used as features in our work. BioNER is the task of extraction of BNEs like gene, protein, chemical etc. from biomedical text. From Figure 1, we can observe that the arguments of the events are BNEs and hence BNEs are useful features for argument identification task.

The combination of these features is used to develop the template feature. The template file sets up which features to use while running CRFs. Each line in the template file represents one template. The template is represented as %x[row,col], where "row" specifies the relative position from the current token and "col" represents the absolute position of the column.

2.3 Experiments

In this work we have followed two step approach, first the event trigger is identified and then the event arguments. One important semantic pre-processing module has been introduced in this work. We have developed Anaphora and Coreference resolution module as part of semantic pre-processing. This is important to resolve the anaphoric entities such as "these proteins", "it" which refer to proteins, chemicals etc. After, incorporating the features, the system was trained with the training corpus. We extracted the distinctive features to build the language models based on conditioned features. Finally, by using these language models, NEs in the testing corpus are automatically labeled. The experiments performed are detailed in this section.

Syntactic Pre-Processing: The syntactic preprocessing of the data is performed using Genia Tagger (Tsuruoka et al., 2005), where the data is split into sentences and tokenized and then PoS and chunk tags are added. The performance of this tool for PoS tagging is 98.26% accuracy and for chunking, the F-score obtained is 88.9% for Noun Phrases and 95.2% for Verb phrases (Kang et al., 2011).

Semantic pre-processing: The semantic preprocessing of the data includes named entity tagging and anaphora resolution. As the event in biomedical text is established between the BNEs, the identification of BNEs is important. Similarly as described in Section 1, resolution of anaphora is an essential step for event extraction that helps in improving the system's performance.

Biomedical Named Entity Recognition: The BNEs are identified using the system developed by (Gopalan et al., 2016). This portable system is developed on three data sets, BioNLP/NLPBA 2004 dataset; BioNLP-ST 2013 (pathway curation task data) and BioCreative 2013 CTD track data using ML approach. A rich feature set including linguistic features and domain-specific features were used to develop the system. For BioNLP-ST corpus they obtained F-score of 83.73%. The BNEs belonging to calss simple chemical, gene or gene product, complex and cellular component are identified. We used this system to identify the named entities from our corpus. Named entities is one of the features used for event argument identification. After identifying the entities, the resolution of anaphora is performed.

Biomedical Anaphora and Co-reference resolution module: Anaphora is a compound word consisting of the words "Ana" and "phora". "Ana" refers to back, upstream or back in an upward direction. "phora" means the act of carrying and denoted the act of carrying back stream. Anaphora is a type of expression whose reference depends upon another referential element. Reference is made based on the preceding part of the utterance. It is the cohesion which points back to some previous items. "The pointing back" is called an anaphor and the entity to which it refers is antecedent. The process of determining the antecedent of anaphor is called as anaphora resolution. Anaphora resolution in discourse is the task or process of identifying the referents of expressions which we use to denote discourse entities, i.e., objects, individuals, properties and relations that have been introduced and talked about in the prior discourse. Biomedical texts differ significantly from other text genres such as newspapers and fiction writing. In biomedical texts, much background knowledge is required for the reader to understand the relation between the entities mentioned in the text. This is a common aspect of scientific papers.

One of the common problems in biomedical texts is a gene and the protein it encodes share the same name, causing some ambiguity in the text when the context does not provide enough information to determine whether the writer is talking about the gene or the protein. Though there are writing conventions to avoid this ambiguity, it is common, however, that authors do not

follow these conventions properly. Other common issue is protein or gene names may coincide with common English words, e.g. for (symbol for foraging). These sources of ambiguity create challenges to a system for automatic detection of entities and events.

The distribution of different types of noun phrases in biomedical articles differs from the distribution in other general text. Pronouns are very rare, accounting for about 3% of noun phrases; whereas proper names, acronyms are very frequent, giving mentions of genes, proteins and names of other BNEs. In the WSJ Newswire corpus the pronouns are 4.5% and in fiction part of the brown corpus the pronouns are 22%. Another aspect in the pronouns distribution in the biomedical texts is it has more plural pronouns such as "these proteins", "them", "those proteins". This makes the task of anaphora and coreference resolution more challenging. It is observed that in biomedical texts entities are commonly referred to using non-pronominal noun phrases, like proper nouns, acronyms or definite descriptions. Hence there is a need to focus on these noun phrases (NPs) for a good event extraction engine. The occurrence of acronyms and NPs which have part-of relationships, linking those in the co-reference chain is a challenging aspect in biomedical coreference resolution.

Though there are many Anaphora and coreference resolutions systems developed for general domain, there are very few works on anaphora and coreference resolution in the biomedical domain. Castano et al., (2002) developed a salience-based system for anaphora resolution which uses UMLS Semantic Network to obtain semantic information. Gaizauskas et al., (2003) developed PASTA system, which implements an inference-based coreference resolution module. Yang et al., (2004) developed a supervised ML approach for anaphora resolution and evaluated it on a portion of the GENIA corpus. Gasperin et al., (2009) developed a statistical anaphora resolution system for biomedical domain. They have tested their system on various corpora.

Here in this work, a general Newswire text anaphora engine is customized to adapt to the biomedical domain. So here anaphora and coreference module is developed using a hybrid approach. An implementation of a general anaphora and coreference resolution engine as described in Lalitha Devi et al., (2011) is done here. The main difference in the implementation is the corpus used for training. Lalitha Devi et al., (2011) use Newswire text documents, whereas in this implementation, the anaphora and coreference annotations provided in BioNLP-ST 2013 GE task have been used. The results obtained from this engine are post processed with rules specific to biological domain. In the post processing stage Genia ontology is used to provide the required world knowledge to resolve the linking of acronyms, for improving the resolution of acronyms and definite descriptions. In Figure 3 the architecture of Anaphora and coreference module is shown in detail.

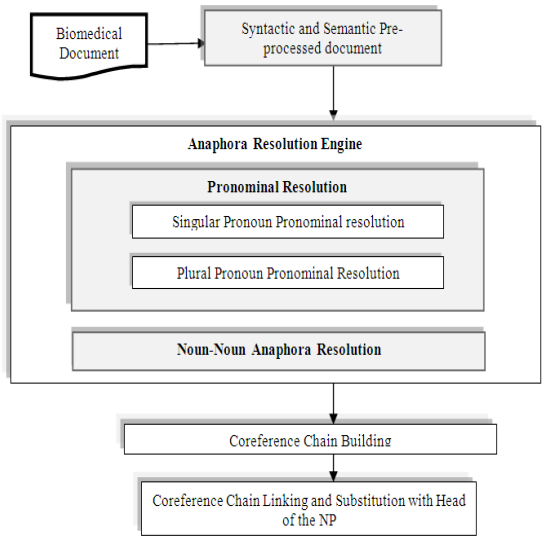


Figure 3: System architecture for Anaphora Resolution

The features used are same as described in the Lalitha Devi et al., (2011). Along with those features, two more features specific to biological domain are added. The new features are

1. Biological Entity type matching: 'yes' if anaphor's and candidate's biological entity type match, 'no' otherwise.
2. Is Entity type a gene or Protein? 'Yes' if the anaphor entity type or candidate entity type is gene or protein, 'no' otherwise. This feature is mainly to distinguish which pairs can hold BNE relations, because most of the event types have arguments as proteins or genes.

This module has been evaluated separately to ascertain its efficiency as a standalone engine. This has been tested on the gold anaphora annotations provided in the test partition of the GE task of the BioNLP 2013 shared task. We have obtained a precision of 55.35%, recall of 58.36% and F-score of 56.86%.

Event extraction: The whole task of event extraction is divided into two sub-tasks. First the event trigger is identified and then the event arguments are extracted. The event extraction sub task includes two phases, event start identifica-

tion and event end identification. We have used the training and development partitions of the GE task data for training and the test partition has been used for testing the system. The experiments performed for event trigger identification and event extraction are described below.

Event Trigger Identification: Event trigger is an important feature for extraction of event from a document. The features used for event trigger identification includes lexico-syntactic features like words, PoS, chunk and morphological patterns. The event trigger for event relation includes noun phrases containing action terms like "regulation", "interaction", "phosphorylation", "expression" etc. In some cases, the event trigger is verb phrases like "activates" that belongs to event type "positive regulation". Hence syntactic features like PoS and chunk acts as prime features for identification of event triggers. In addition to lexical and syntactic features, we have also used another biomedical domain specific feature, "trigger indicator". Trigger indicator feature includes biomedical domain specific verbs such as "binds", "inhibit" etc. and biomedical key terms like "translocation", "methylation" etc. This feature has a Boolean value "true" if domain specific verbs or key terms are identified in the current word, else "false". For event trigger identification the data is first preprocessed and features are extracted. After extracting the features, the language model is built. The identification of event trigger is followed by the identification of event arguments.

Event Argument Identification: The event extraction task includes extraction of event and its arguments. For extraction of event arguments, the event start and event end is identified. The event trigger is identified in the first task and the sentences with event trigger are given as input to the event argument extraction module. The features for event boundary identification are word, PoS, chunk, event trigger and BNEs. The arguments for the events are BNEs. Hence, giving weightage to BNEs that occur before or after the event trigger helps in the identification of argument boundaries. Using these features the language models are built for event start boundary and event end boundary. These models are used for identifying and extracting the event of a text.

The event may have one argument or multiple arguments. In case of events like "gene expression", there will be one argument "theme". Whereas in case of events like "binding" and "regulation", there will be more than one arguments such as "theme", "cause" and "site". Consider the Example 1 given below.

Example 1:

Methyl-CpG-binding proteins (MBPs) are thought to **inhibit** the **binding** of ***transcriptional factors*** to the ***promoter***.

In this Example 1 there are two events "negative regulation" and "binding". The event triggers are "inhibit" for negative regulation and "binding" for binding event. The arguments for "negative regulation" event is the theme "binding of transcriptional factors", which again is an event, the cause "Methyl-CpG-binding proteins" and the site "promoter". This event has three arguments. The second event is "binding" and the arguments of this event are the theme "transcriptional factors" and the site "promoter". For this event there are two arguments. The simpler events mostly have one argument.

From this example we also observe that the arguments of first event and second events are overlapping and importantly the second event as a whole is one of the argument of first event. Both the events share same arguments and hence the argument boundaries overlap. In these cases we have processed the events separately i.e. when there is more than one event in a sentence; each event is processed one by one, while developing the models.

The event arguments are actually relations between the entities. Thus the event argument identification is modeled as the identification of argument spans for each argument of the event trigger. The basic assumption is that each event will either have an explicit or an implicit event trigger. Event argument span identification is split into four sub-phases for identification of each boundary of each argument, i.e., the identification of Arg1's two boundaries and Arg2's two boundaries. Four language models were built for this purpose and Arg2-START, Arg1-END, Arg1-START and Arg2-END were identified in series, in that order. The output at each sub-phase was fed as input to the next sub-phase. In other words, in each sub-phase, the previously identified boundary is also used as a feature along with the features explained in Section 2.2. The choice of the order of identification of bounds was made with the idea that it is easier to first find the boundaries that are in close proximity to the cause-effect marker – Arg1-END and Arg2-START. Between these two, Arg2-START was chosen first, arbitrarily. The same holds for the choice of Arg1-START to be the third boundary. The arguments need not be always adjacent to

the marker. Sometimes, the arguments can be in the same sentence as the event trigger as shown in Example 1. Sometimes, one of the arguments is in the sentence immediately preceding that of the event trigger.

3 Results and Discussion

This section describes the performance of our system in terms of Precision, Recall and F score. Precision is the number of NEs correctly perceived by the system from the total number of NEs identified, Recall is the number of NEs correctly detected by the system by the total number of NEs contained in the input text and F-score is merely the harmonic mean of precision and recall.

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F score = (2 × Recall × Precision / (Recall + Precision))

Where, TP means true positives, FN means false negatives and FP means false positives.

We evaluated our system on test partition of GE task data. The overall result and results achieved by the system for each event type are demonstrated in Table 2 and Table 3. First, we developed the system for event extraction, without resolving the anaphors. Then we improved the performance of the system by resolving the anaphors. We obtained F-score of 75.12% for simple events, 62.11% for Binding Events & Protein Modification Events, 35.31% for regulation events and 49.27% for all event types without resolving the anaphors. After resolving the anaphors we observed that there is a significant increase in the performance of the system for identification of binding and modification events and regulation events.

Event Types	Precision	Recall	F-score
Simple Events	78.65	71.89	75.12
Binding & Protein Modification Events	66.36	58.54	62.11
Regulation Events	41.43	30.77	35.31
Overall	60.15	41.73	49.27

Table 2: Results for event extraction- Without Anaphora & Coreference resolution

Event type	Precision	Recall	F-score
Simple Events	78.75	71.94	75.76
Binding & Protein Modification Events	69.87	61.67	65.51
Regulation Events	46.15	35.42	40.08
Overall	67.15	47.73	54.25

Table 3: Results for event extraction-After Anaphora & Coreference resolution

Although simpler events achieve good results in event extraction task, the extraction of events such as binding and modification events and regulation events is still difficult. We have made an approach to improve the results of these complex events by resolving anaphora and co-reference. There are 445 anaphora relations in binding and regulation event types. With the help of our anaphora resolution engine we were able to identify the referents of 254 anaphor relations correctly. The anaphor relation consisted of pronominal anaphors such as them, its, they etc. and noun-noun anaphors such as "aforementioned cytokines", these proteins" etc. Consider the below Example 2

Example 2

After 5 days, supernatants were collected and the secretion of ***IFNgamma, IL4 and IL2*** were measured by ELISA. Samples from both negative controls had no detectable production of the ***aforementioned cytokines***.

In Example 2, the event trigger is "production" and the event is "gene expression". The argument for the event is "aforementioned cytokines", but this refers to IFNgamma, IL4 and IL2. This is an example for noun-noun anaphora relation. The main objective of this task is to identify the protein involved in the event. If we do not resolve the noun-noun anaphor "aforementioned cytokines", we will not be able to identify the protein names. Hence resolving the anaphors helped in the improvement of regulation and binding events. To know the significance of each features we conducted experiments to check the performance of individual features. The results for performance of individual features are shown in Table 4.

Table 4: Results for Individual features

Features	Precision	Recall	F-score
Lexical feature	37.87	23.13	28.53
Lexical feature +Syntactic features	49.46	28.79	36.80
Lexical feature +Syntactic features +Event trigger	60.56	37.01	45.94
Lexical feature +Syntactic features +Event trigger + BNEs	60.45	41.65	49.31
All above features + Anaphora	67.15	47.73	54.25

For event extraction we have used lexical feature, syntactic features such as PoS and chunk, event trigger and biomedical entities. When lexicon is used as feature we obtained precision of

37.87%, recall of 23.13% and 28.53% F-score. A window size of 5 is used. Then we used syntactic features along with the lexical feature. Since PoS and chunk plays a key role in extraction of event trigger and arguments, we observed that there were significant improvement in precision and recall of the system. There was a significant increase in F-score of about 8.27%. Then we used event trigger as feature along with lexical and syntactic feature. Event trigger is very important feature in event extraction task as it signals the presence of event in a text. We obtained good increase in performance with precision of 60.56%., recall of 37.01% and F-score of 45.94%. Then we used BNE features along with the other features. BNEs are main features for argument identification of an event. This feature helped in heightening the system's performance with increase in F-score of about 3.37%. We obtained an increase of 4.94% after resolving the anaphors. The evaluation results show that the system is comparable to state of art system.

4 Conclusion

This paper described an event extraction system designed using the ML approach CRFs, with rich feature set. We have evaluated our system on test partition of GE task data and showed that the system is comparable to state of art system. The performance of the system based on individual feature is outlined and have also exhibited that the system render good performance by resolving the anaphors.

Reference

John Lafferty, Andrew McCallum and Fernando C. N. Pereira. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data, *Proceedings of the Eighteenth International Conference on Machine Learning*, San Francisco, USA, 282–289.

Kudo Taku. 2005. *CRF++, An Open Source Toolkit for CRF* [online] http://crfpp.sourceforge.net (accessed 3 January 2013).

Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo and Pierre Zweigenbaum. 2013. Overview of BioNLP Shared Task 2013, *Proceedings of the BioNLP Shared Task 2013 Workshop*, Sofia, Bulgaria, 1–7.

Sindhuja Gopalan and Sobha L. Devi. 2016. BNE-Miner: mining biomedical literature for extraction of biological target, disease and chemical entities, *Int. J. Business Intelligence and Data Mining*, 11(2):190–204.

Bairoch, A., Apweiler, R., Wu, C.H., Barker, W.C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., Martin, M.J., Natale, D.A., O'Donovan, C., Redaschi, N., and Yeh. L.S.L. (2005) 'The universal protein resource (uniprot)', *Nucleic Acids Research*, 33(1):154–159.

Samuel Kerrien, Bruno Aranda, Lionel Breuza, Alan Bridge, Fiona Broackes-Carter, Carol Chen, Margaret Duesbury, Marine Dumousseau, Marc Feuermann, Ursula Hinz, Christine Jandrasits, Rafael C. Jimenez, Jyoti Khadake, Usha Mahadevan, Patrick Masson, Ivo Pedruzzi, Eric Pfeiffenberger, Pablo Porras, Arathi Raghunath, Bernd Roechert, Sandra Orchard and Henning Hermjakob. 2012. The intact molecular interaction database in 2012, *Nucleic Acids Research*, 40(1):841–846.

Karin Verspoor, Kevin B. Cohen, Arrick Lanfranchi, Colin Warner, Helen L. Johnson, Christophe Roeder, Jinho D. Choi, Christopher Funk, Yuriy Malenkiy, Miriam Eckert, Nianwen Xue, William A. Baumgartner, Michael Bada, Martha Palmer and Lawrence Hunter. 2012. A corpus of full-text journal articles is a robust evaluation tool for revealing differences in performance of biomedical natural language processing tools, *BMC Bioinformatics*, 13(1): 207.

Jose Castano, Jason Zhang and James Pustejovsky. 2002. Anaphora resolution in biomedical literature, *Proceedings of International Symposium on Reference Resolution for NLP 2002*, Alicante, Spain.

Robert Gaizauskas, Demetriou, G., Artymiuk, P.J., and Willett, P. (2003) Protein structures and information extraction from biological texts: the PASTA system, *Bioinformatics*, 9(1):135-143.

Xiaofeng Yang, Jian Su, Guodong Zhou and Chew L. Tan. 2004. An NP-cluster based approach to coreference resolution, P*roceedings of COLING 2004*, Geneva, Switzerland, 226–232.

Jari Bjorne and Tapio Salakoski. 2013. TEES 2.1: Automated Annotation Scheme Learning in the BioNLP 2013 Shared Task, *Proceedings of the BioNLP Shared Task 2013 Workshop*, Sofia, Bulgaria, 16–25.

Kai Hakala, Sofie V. Landeghem, Tapio Salakoski, Yves Van de Peer and Filip Ginter. 2013. EVEX in ST'13: Application of a large-scale text mining resource to event extraction and network construction, *Proceedings of BioNLP Shared Task 2013 Workshop*, Sofia, Bulgaria, 26-34.

Sobha L. Devi, Pattabhi R. K. Rao, Vijay S. Ram, Malarkodi C. S, and Akilandeswari A. 2011. Hybrid Approach for Coreference Resolution, *Proceedings of 15th Conference on Computational Natural Language Learning: Shared Task*, Portland, Oregon, 93-96.

Sofie V. Landeghem, Filip Ginter, Yves Van de Peer and Tapio Salakoski. 2011. Evex: A pubmed-scale resource for homology-based generalization of text mining predictions, *Proceedings of BioNLP 2011 Workshop*, Portland, Oregon, USA, 28–37.

Caroline V. Gasperin. 2009. Statistical anaphora resolution in biomedical texts. A Technical report based on a dissertation titled Statistical anaphora resolution in biomedical texts, University of Cambridge.

Ning Kang, Erik M. van Mulligen and Jan A. Kors. 2011. Comparing and combining chunkers of biomedical text, *Journal of Biomedical Informatics*, 44(2):354–360.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou and Jun'ichi Tsujii. 2005. Developing a Robust Part-of-Speech Tagger for Biomedical Text, *Lec-*

Co-reference Resolution in Tamil Text

Vijay Sundar Ram R. and Sobha Lalitha Devi
AU-KBC Research Centre,
MIT Campus of Anna University,
Chennai, India
{sundar,sobha}@au-kbc.org

Abstract

Natural Languages are cohesive. Cohesiveness is brought by various language phenomenons. Co-referring entities bind the sentence through reference phenomenon. These co-referring entities include various anaphoric expressions namely pronominals, reflexives, reciprocal, distributives, noun-noun anaphora and definite descriptions. These co-referring entities form the co-reference chains. In this work, we present a methodology to identify the co-reference chains in Tamil text. Evaluation of the system shows encouraging results.

1 Introduction

Cohesiveness of the text is brought by various language phenomenons. Co-referring entities play a crucial part in binding discourse intra and inter sententially. These co-referring entities form a chain in the text. The present work is on identifying the co-reference chains in Tamil text by identifying the co-referring entities. The co-referring entities consist of pronominal, reciprocal, reflexives, distributives, definite description and noun-noun anaphora and their antecedents. Co-reference chains are very essential in building cutting edge natural language processing tools such as profile building, entity based summary generator, entity specific sentiment analyser etc. Consider the following the discourse.

Ex. 1.a
***raaju*$_{1,2}$ *avanutaiya*$_1$** *naNpan* ***baaluvin*$_2$**
Ramu(N) he(PN)+gen friend(N) Balu(N)-gen
viittiRku *cenRaan.*
house(N)+dat go(V)+past+3sm
(Raju went to his friend Baalu's house.)

Ex. 1.b
***ivarkaL*$_{2,3}$** ***oruvarukkoruvar*$_3$** *nanku*
They(PN) eachother(N) very_well(ADJ)
aRivaarkaL. (1.1.b)
know(V)+past+3p
(They know each-other very well.)

Ex. 1.c
baaluvin *thaay* ***siiththaa*$_4$**
Balu(N)-gen mother(N) Sita(N)
oru *aaciriyar.* (1.1.c)
one(ADJ) teacher(N)
(Baalu's mother Sita is a teacher.)

Ex. 1.d
baalu*$_4$** ***raajuvai1 ***than*$_4$** ***naNparkaL*$_5$**
Balu(N) Raju(N)+acc his(PN) friends(N)
ovvoruvarukkum$_5$ aRimukappatuththinaan.
everone(PN) introduced(V)
(Baalu introduced Raju to every-one of his friends.)

There are various anaphoric expressions in the above discourse (Ex.1) and following are the pronominals, 1.'avanutaiya' [his] refers to 'raaju' (Ex.1.a), 2. 'than' [his] (Ex.1.d) refers to 'baalu' (Ex.1.d), 'ivarkaL' [they] in (Ex.1.b) refers to 'raaju' and 'baalu' present in (Ex.1.a) as two independent mentions. Here the antecedent of the pronoun 'ivarkaL' is two independent nouns 'raaju' and 'baalu'. This type of antecedents is known as split-antecedents.

The reciprocal 'oruvarukkoruvar' [each-other] in Ex.1.b refers to 'ivarkaL' [they], the subject of the sentence. In Ex.1.f, 'ovvoruvar' [every-one], Distributive refers to 'naNparkaL' (friends). The other type of anaphor is the noun-noun anaphor which is present in most of the sentences.

This can be seen in the above exampla as 'baalu' in Ex.1.d refers to 'baalu' in the previous sentence, 'baaluvin' in Ex.1.a. Similarly 'raajuvai' in Ex.1.d refers to 'raaju' in Ex.1.1.a. From

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 392–401

the above explanation we can form the co-reference chain for each type of reference and they are given below.

Following are the Co-reference Chains from the example sentences from Ex.1.a to Ex.1.d.

- '**raaju**' (Ex.1.a) , '**avanutaiya**' [his] (Ex.1.a), '**raajuvai**' (Ex.1.d)
- '**naNparkaL**' (Ex.1.d), '**ovvoruvar'** [every-one] (Ex.1.d)
- '**raaju**', '**baalu**' (Ex.1.a), '**ivarkaL**' [they] (Ex.1.b), '**oruvarukkoruvar'** [each-other] (Ex.1.b)

Thus the anaphoric expressions such as Pronominal, Split-antecedents, Reciprocal, Reflexives, Distributive, One anaphora, Definite-Descriptions and Noun-Noun anaphora constitute the co-reference chains. In the present work, the co-reference chains are built by resolving these anaphoric entities.

Co-reference resolution was the shared task in DARPA's Message Understanding Coreference MUC-6 (1995) and MUC-7 (1997). These two shared tasks were the early initiatives which kick started machine learning based approach for co-reference relation resolution task. Aone & Bennet (1995), McCharthy & Lehnert (1995), Fisher et al. (1995) had used decision tree learning algorithm to come up with co-reference resolution system. . Aone & Bennet (1995) demonstrated the system with Japanese texts along with English texts. Kelher et al. (1997) used maximum entropy modelling technique to built co-reference resolution engine. Cardie & Wagstaff (1999) came up with an un-supervised learning approach to identify co-reference relation. They have evaluated their engine on MUC-6 dataset. Soon et al. (2000) used decision tree learning approach to identify the co-referencing pairs and used pair-wise model to build the co-reference chains. Ng & Cardie (2002) enhanced Soon et al. (2000) decision tree learning approach with more linguistic and heuristic features. They used best-first clustering methodology to build the co-reference chains. First-order probabilistic model was by Culcotta et al. (2007). Bengston and Roth (2007) tried to present that the approach by Soon et al. (2000) would perform better with better features. They re-implement it with modified features. Rahman & Ng (2011) employed cluster-ranking approach to perform co-reference resolution. A multilevel sieve based approach was performed by Raghunathan et al. (2010). SemEval (2010) Coreference Resolution in Multiple Languages aimed to explore the portability of systems across languages, need for different levels of linguistic information (Recasens, 2010).

The flow of the paper is as follows. In the following section, we present a brief introduction on characteristics of Tamil. We have explained our approach in the third section. In fourth section, we have presented on the experiment, result and observation. The paper concludes with the conclusion section.

2 Characteristics of Tamil

Tamil belongs to the South Dravidian family of languages. It is a verb final language and allows scrambling. It has post-positions, the genitive precedes the head noun in the genitive phrase and the complementizer follows the embedded clause. Adjective, participial adjectives and free relatives precede the head noun. It is a nominative-accusative language like the other Dravidian languages. The subject of a Tamil sentence is mostly nominative, although there are constructions with certain verbs that require dative subjects. Tamil has Person, Number and Gender (PNG) agreement. It is a relatively free word order language, but when it comes to noun phrases and clausal constructions it behaves as a fixed word order language. Clausal constructions are introduced by non-finite verbs. Tamil has copula drop, accusative drop, genitive drop, and PRO drop (subject drop).

3 Our approach

In this section, we present our approach to identify the co-reference chain in Tamil text. In most of the published works, single machine learning technique with a set of features is used. We have varied from other approaches by using different methodologies and features for resolution of various anaphoric expressions as resolution of pronominals, reciprocal, reflexives, distributives requires syntactic features and resolution noun-noun anaphora and definite description requires semantic features.

Our approach starts with preprocessing input text with sentence splitter, tokeniser and syntactic modules namely morphological analyser built using paradigm based approach (Sobha et. al, 2013), PoS tagger (Sobha et. al, 2016) and chunker using Conditional Random Fields (CRFs) technique, and clause boundary identifier built using CRFs with linguistic rules as features (Ram et. al., 2012) and Named Entity recognizer built using CRFs where statistical features are used (Malarkodi et. al., 2012).

The preprocessed text is fed to various anaphora resolution engines, pronominal resolution engine, where the singular pronouns are resolved using ML techniques and plural pronouns are resolved using salient weights based approach; reflexives, reciprocals and distributives are resolved using rule based approach; followed by noun-noun anaphora resolution and definite description identification using CRFS techniques. Using the different anaphors and their antecedents, co-reference chains are built using pair-wise clustering techniques with restriction rules. We have described the methodologies of resolution of various anaphoric expressions in the following sub-section.

3.1 Pronominal Resolution

We have performed the resolution of singular and plural pronouns with different techniques as plural pronouns can have plural noun phrase, co-ordinated noun phrases and split-antecedents.

Singular Pronoun Resolution: Singular pronoun resolution is built using Conditional Random Fields (CRFs) technique (Kudo,2013). Though CRFs is notable for sequence labelling task, we used this technique to classify the correct anaphor-antecedent pair from the possible candidate NP pairs by presenting the features of the NP pair and by avoiding the transition probability. While training we form positive pairs by pairing anaphoric pronoun and correct antecedent NP and negative pairs by pairing anaphoric pronouns and other NPs which match in person, number and gender (PNG) information and match Named entities (NE) constraints with the anaphoric pronoun. NE constraints check for the type of NE which can be the antecedent for a particular pronoun, such person pronoun can have Individual as antecedent and Location NE can never be its antecedent. These positive and negative pairs are fed to the CRFs engine and the language model is generated. While testing, when an anaphoric pronoun occurs in the sentence, the noun phrases which match in PNG and satisfies NE constraints with the pronoun, that occur in the preceding portion of the sentence and the four preceding sentences are collected and paired with the anaphoric pronoun and presented to CRFs engine to identify the correct anaphor-antecedent pair.

The features used in machine leaning technique are as follows.

Positional Features: Is the candidate antecedent occur in the same sentence where the pronoun has occurred or in the prior sentences.

Syntactic Argument: The case marker affixed to the NP helps in identifying the systactic argument of the sentence such as subject, object, indirect object, are obtained from the case suffix affixed with the noun phrase. The case marker information is available from the morphological analysers output.

Linguistic Characteristics:

a) PoS tag and chunk information of Candidate NP, suffixes affixed with the noun.
b) The suffixes which show the gender which gets attached to the verb.
c) Whether the candidate NP (probable antecedent) is Possessive.

Constraint Features: The constraint features are obtained from clause boundary information. If the pronoun is possessive, the nominative NP within the clause has a high probability to be the antecedent. If the pronoun is a non-possessive pronoun, the nominative NP in the immediate preceding clause has a high probability to be the antecedent. So we check the position of the candidate NP with respect to clause boundary such as whether the candidate NP occurs in current clause or immediate clause or non-immediate clause.

Plural Pronoun Resolution: Plural pronoun resolution engine is developed using a salience factor weights based approach. The antecedent for a plural pronoun can be a plural Noun phrase, co-ordinated NPs and Split antecedent.

We weigh each of the Noun phrase matching in gender with the plural pronoun. The features for the salience factors are obtained from the syntactic parsing output. We have mentioned the salience factors and its weights were as per Sobha (2007). Following is the algorithm used in resolving plural pronouns.

Step 1: If a plural pronoun occurs then Step 2.
Step 2: Collect all Noun phrases in the current sentence and previous four sentences which match with the gender of the plural pronoun.
Step 3: Each Noun phrase (NP) in the collection of possible antecedent set is scored with salience factor weights.
Step 4: The NPs re-sorted in descending order with their weights.

Step 5: If the highest scored NP is a plural NP, then it is selected as the Antecedent. Else step 6.
Step 6: If the highest scored NP is singular, check if this NP is part of co-ordinated NP or split antecedent, then choose the co-ordinated NP or the split antecedent as the antecedent.

Check for Co-ordinated NP**:** Co-ordinated NPs are those NPs which have the same scores as the highest score NP.

Check for Split-antecedents: We attempt to identify split-antecedents using selectional restriction rules of the verb, categorizing the nouns based on its sub-categorization information and ranking the possible antecedents using salience weights.

Sub-categorization features explain the nature of a noun. Sub-categorization information includes the features such as [±animate], [±concrete], [±edible] etc. The verbs describe the action or the process in the nature and this allow the verbs to take nouns with specific sub-categorization feature as its syntactic arguments. This is defined as the selectional restriction rules of a verb.

Ex.2

raam aappil caappittaan.

Ram(N) apple(N) eat(V)+past+3sn

(Ram ate an apple).

Here in Ex.2 'raam' (Ram) has the sub-categorization feature [+animate, +human] and 'aappil' (apple) with [+edible]. The selectional restriction features required by the verb 'caappitu' (eat) for selecting its subject and object are [+animate] and [+edible] respectively. If there is a violation in SR rules, the sentence can be syntactically correct but it will not be semantically correct. Verb has the right to select its arguments. We have grouped the verbs according to the sub-categorization information of the subject and object nouns. A group of commonly used 1500 verb senses are analyzed and 500 selectional restriction rules are derived by (Ananth and Sobha, 2010). The sub-categorization features of a noun are explained in the next section. A sample rule is shown in fig 1.

Using the selectional restriction rules and the sub-categorization information of nouns we try to group the noun phrases to form groups which can be possible split-antecedents.

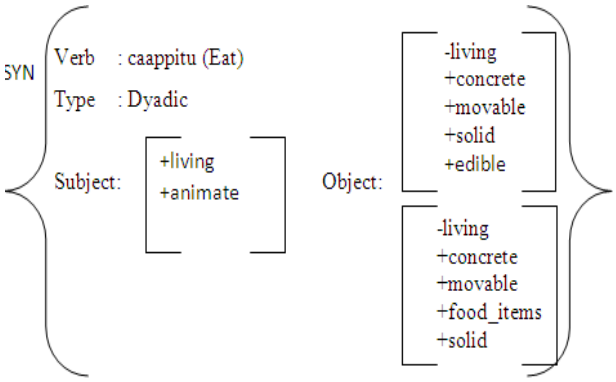


Figure 1. Selectional restriction rule for 'caapitu' (eat).

Following are the steps involved in identifying the split-antecedents. In the first step, we enrich the nouns and the verbs with its sub-categorization information, and selectional restriction rules respectively. The named entities (NEs) are mapped to the sub-categorization features, so we get the sub-categorization information using the NE information as described in the example Ex.3 and Ex.4.

Ex.3.

Person: [+living; +animate; +vertebrate; +mammal; +human;]

Ex.4.

Location: [-living; -moveable; +landscape]

In the second step, when a plural pronoun is encountered in the sentence, the preceding portion of the sentence and two preceding sentences are considered for analysis, as Gatt et al. (2009) has shown that the distance between plural pronouns and its antecedent are very few sentences away. The noun phrases in the preceding sentences are analysed and grouped to form the possible antecedents. For grouping the NPs, the NPs need to satisfy the following matching conditions.

a) The NPs can be grouped together if they have same sub-categorization information or till the last but one node in the ontology is same. Example [+living; +animate; +vertebrate; +mammal; +human; +female] and [+living; +animate; +vertebrate; +mammal; +human; -female] are considered to be same since both are same till last but one node and the exceptions are as follows:

In the case of NPs with sub-categorization [+living] and do not have [+human], we look for sub-categorization match between the NPs only

till [+living; +animate] and such NPs are grouped together.

Following are the steps involved to form possible candidates by grouping the NPs.

1. Identify the plural pronoun in n^{th} sentence.
2. If the finite verb of the sentence having plural noun or plural possessive pronoun is followed by noun form of the verbs such as 'inai' (merge), 'manam' (marry), 'vivaakaraththu_cey' (disvorce), 'kaathal' (love) then look for two nouns which satisfy the sub-categorisation matching condition in preceding two sentences, group these two NPs as a possible split antecedent candidate.
3. Consider sentence n-2^{th}, n-1^{th} and in n^{th} sentence consider the portion preceding to the plural pronoun to form a candidate sentence set.
4. For each sentence in the candidate sentence set
    a. Noun Phrases with conjunct suffix 'um' or conjunct word 'maRRum' (and) are united to form conjunct NPs.
5. For each sentence in sentence set
    a. If there exists NPs satisfying the matching condition, then the NPs are grouped together.
6. Group the NPs that occur in same syntactic argument position and satisfy the matching condition across n^{th}, n-1^{th} and n-2^{th} sentences.

Table 1 Salience Factors and its Weights

S. No.	Salience Factors	Weights
1	Same Ontology Nodes	30
2	NPs with following verbs	30
3	NPs with same syntactic argument position	20
4	NPs with different syntactic argument position	10
5	NPs are syntactic argument for verbs having same SR rules	30
6	NPs are syntactic argument for verbs with different SR rules	10
7	NPs in current n^{th} sentence	30
8	NPs in n-1th sentence	20
9	NPs in n-2th sentence	10

In the third step, when the possible antecedents are formed by grouping the NPs, they are ranked based on the salience factors derived from the features of NPs such as the sub-categorization information of NPs, the SR rules of verbs followed by the NPs and the syntactic argument position of the NPs in the sentences. The salience factor weights are described in table 1. The weights for the salience factors were initially manually assigned based on linguistic considerations and fine-tuned through experiments (Ram & Sobha, 2016).

Resolution of Reflexives: The antecedent of the reflexive is always the subject of the clause, where the reflexive occur. So the antecedent of the reflexive is identified with the rule based approach.

Resolution of Reciprocals and Distributives: Reciprocals and Distributives are handled similar to the reflexives. The antecedent of the Reciprocals and Distributives will the plural nominative noun phrase in the same clause. The resolution of the reciprocals and distributives are done using a rule based approach.

Noun-Noun Anaphora Resolution: Noun-Noun Anaphora resolution is the task of identifying the referent of the noun which has occurred earlier in the document. In a text, a noun phrase may be repeated as a full noun phrase, partial noun phrase, acronym, or semantically close concepts such as synonyms or superordinates. The engine to resolve the noun anaphora is built using Conditional Random Fields technique. Features used in Noun-Noun Anaphora Resolution are discussed below.

We consider the noun anaphor as NP_i and the possible antecedent as NP_j. Unlike pronominal resolution, Noun-Noun anaphora resolution requires features such as similarity between NP_i and NP_j.We consider word, head of the noun phrase, named entity tag and definite description tag, gender, sentence position of the NPs and the distance between the sentences with NP_i and NP_j as features.

Features used for ML

The features used in the CRFs techniques are presented below. The features are divided into two types.

Individual Features:

1. Single Word: Is NP_i a single word; Is NP_j a single word
2. Multiple Words: Number of Words in NP_i; Number of Words in NPj
3. PoS Tags: PoS tags of both NP_i and NP_j.
4. Case Marker: Case marker of both NP_i and NP_j.

5. Presence of Demonstrative Pronoun: Check for presence of Demonstrative pronoun in NP_i and NP_j.

Comparison Features

1. Full String Match: Check the root words of both the noun phrase NP_i and NP_j are same.
2. Partial String Match: In multi world NPs, calculate the percentage of commonality between the root words of NP_i and NP_j.
3. First Word Match: Check for the root word of the first word of both the NP_i and NP_j are same.
4. Last Word Match: Check for the root word of last word of both the NP_i and NP_j are same.
5. Last Word Match with first Word is a demonstrator: If the root word of the last word is same and if there is a demonstrative pronoun as the first word.
6. Acronym of Other: Check NP_i is an acronym of NP_j and vice-versa.

Definite Description Identification: Definite Description (DD) is a unique denoting phrase of an entity. Consider the example, Indian Prime Minister Narendra Modi. Here the phrase "Indian Prime Minister" describes about Person Entity 'Narendra Modi'.

We used CRFs technique to identify the DD relations. We have used the PoS, NE information of the two NPs (possible definite description NP and Entity NP) and two preceding and following words as the feature to train the CRFs engine.

Co-reference Chain Builder: We used CRFs technique to identify the DD relations. We have used the PoS, NE information of the two NPs (possible definite description NP and Entity NP) and two preceding and following words as the feature to train the CRFs engine. We have used various constraint rules to generate the co-reference chains from the co-referring antecedent NP and anaphor NP pairs. We have built the constraint rules based on the types of the NPs in the co-referring NP pairs. Co-referring pairs obtained from different pronominal resolution engines are treated with high confidence. Co-referring NPs having exact match and not a partial NPs of any other NP, then these pair of NPs are considered for generating co-reference chains. If one of the NPs in the co-referring NP pair is a definite description, then the distance between should be checked. If it is close by in the same sentence then it is considered for co-reference chain generation. If one of the NP is a partial NP in the pair, then the distance between the partial NP and its co-referring NP is checked. If the distance is more than 3 sentences then the pair is dropped. If both the NPs are partial NPs and if the antecedent NP has a co-referring NP within proceeding three sentences then we can consider the pair for co-reference chain generation. The algorithm is presented as follows.

Algorithm for Generating Co-reference Chains

Step1: Type of NP in each co-referring NP pairs are identified.

Step2: For each of the identified co-referring NP pair; do step3 to

Step3: Check for the types of NPs in the co-referring NP pair,

If both the NPs have exact match and not a partial NP of the full NP then do step4.

If co-referring pair is obtained from the pronominal resolution engines, then do step4.

If one of the NP is a definite Description in the NP pair, check if the NPs occur close in the same sentence, then do step 4.

If the pair of NPs has a full NP and a partial NP, check if the NPs are in close proximity, i.e. within the three preceding sentences, then do step4.

If the pairs of NPs have both partial NPs, then check if the antecedent NP has a co-referring NP in the preceding three sentences, then do step4.

Step 4: Check if the NPs in the co-referring NPs are part of one of the existing clusters of co-referring NPs, then include these two pairs in that cluster. Else, introduce a new cluster with these two NPs.

Step 5: Each cluster is formed into a co-reference chain.

4 Experiment, Results and Discussion

We have manually annotated 1000 Tamil newwires collected from online Tamil web pages belonging to three domains, viz, sports, general and disaster. We had two annotators and the inter-agreement score is measured to be 0.78 kappa score. We have used 80% of the annotated corpus for developing the different anaphora resolution engines and co-reference chain builder. The rest 20% of the annotated corpus is used for testing the different anaphora resolution engines.

In the following table 2, we have presented the statistics of the annotated corpus.

Table 2: Basic Corpus Statistics

Details about Corpus	Count
Number of Web Articles annotated	1,000
Number of Sentences	22,382
Number of Tokens	272,415
Number of Words	227,615

Following table 3, has the statistics of the different anaphoric expression annotated in the corpus.

Table 3: Statistics of Anaphoric expressions in the Corpus

S.No	Type	Number of Occurrence
1	Noun-Noun Anaphora	11,935
2	Anaphoric Pronominal	4,160
3	Definite-Description	1,890
4	Reflexives	29
5	Reciprocal	31
6	Plural pronouns with split-antecedent	190
7	Distributives	8
	Total	18,243

The co-reference chains are evaluated with standard evaluation metrics such as MUC, B-Cubed, CEAFe, CEAFm and BLANC. The performance scores for co-reference chain identification are presented in table 4.

Table 4: Performance scores for Co-reference chains

S.No.	Metric	Precision (%)	Recall (%)	F-Measure (%)
1	MUC	51.21	35.5	41.94
2	B-CUB	74.8	52.71	61.84
3	CEAFm	46.31	46.31	46.31
4	CEAFe	30.2	44.73	36.06
5	BLANC	64.35	56.74	57.80
6	Average	53.37	47.19	48.79

The performance scores of various anaphora resolution modules with system preprocessed corpus and gold standard corpus as input is presented in table 5.

Table 5 presents the comparison of performance scores between the results obtained by giving preprocessed corpus, Gold standard and system processed, as input to the anaphoric systems. This brings out the inherent errors of each anaphora resolution systems and the errors introduced by preprocessing modules. On analysing the gold standard corpus result, we find pronominal resolution and one-anaphora resolution need improvement at the anaphora analysis level. The tendency of pronominal resolution engine to choose the first nominative as antecedent is one of the reason and this needs further analysis.

Table 5 Comparison of Results with System Preprocessed Corpus and Gold standard Corpus as Input

S. No	Task	System Preprocessed Corpus			Gold Standard Corpus		
		Precision (%)	Recall (%)	F-Measure (%)	Precision (%)	Recall (%)	F-Measure (%)
1	Singular Pronoun Resolution	79.04	62.87	70.03	81.63	75.39	78.38
2	Plural Pronoun	81.41	64.7	72.09	82.15	76.21	79.06
3	Reflexives	96.54	93.34	94.91	96.54	93.34	94.91
4	Reciprocals	98.17	97.39	97.78	98.17	97.39	97.78
5	Distributives	97.38	95.56	96.46	97.38	95.56	96.46

5	Definite-Description	92.98	70	79.87	93.83	78.56	85.51
6	Noun-Noun Anaphora Resolution	86.14	66.67	75.16	87.19	78.32	82.52

Table 6 Percentage Distribution of Errors introduced by Various Preprocessing Modules

S. No	Task	Intrinsic Errors of the anaphoric modules (%)	Total Percentage (%) of Error introduced by Preprocessing modules	Percentage of error contributed by Each Preprocessing module				
				Anaphoric Non anaphoric Identification (%)	Morphological Analyser (%)	PoS Tagger (%)	Chunker (%)	Named Entity Recogniser (%)
1	Singular Pronoun Resolution	21.62	8.35	10.86	26.14	40.65	22.35	
2	Plural Pronoun	20.94	6.97	12.56	27.44	37.23	22.77	
3	Reflexives	5.09	0		23	41.66	35.34	
4	Reciprocals	2.22	0		41.45	32.15	26.40	
5	Distributives	3.54	0		38.56	28.14	33.30	
6	Definite-Description	14.49	5.64			25.24	30.27	44.49
7	Noun-Noun Anaphora Resolution	17.48	7.36		11.56	18.78	36.44	33.22

In table 6, we have presented the percentage of intrinsic errors, the total percentage of errors introduced by preprocessing modules to each anaphora resolution engine and the percentage of errors contributed by each preprocessing modules to the total preprocessing errors.

With the informations from table 6, we can understand the importance of features derived from each preprocessing module for developing various anaphora resolution engines.

The output from the gold standard corpus as input is analysed and the observations are discussed below.

In singular pronominal resolution engine, which is built using CRFs techniques, the first nominative NP is choosen as the antecedent if the sentences have more than one nominative NP. Consider the following discourse.

Ex. 5.a.
munnaal thalaivar coomuvin
Formar(N) leader(N) Soomu(N)+pos
aatharavaalaraana raamu
supporter(N) Ramu(N)
neeRRu peecinaar.
yesterday(Adv) talk(V)+past+3sh

Ex.5.b.
avar kuuRiyathu.
He(PN) say(V)+past+3sn

The antecedent for ‘avar’ 3rd person singular honorofic pronoun in Ex.5.b is ‘raamu’ (Ramu) in Ex.5.a. But the resolution engine identifies ‘munnaal thalaivar’ (former leader) as the antecedent. This is also observed in plural pronoun resolution engine. Consider the following discourse.

Ex.6.a
puunaikaL miinkaL neeRRu caappittana.
Cat(N)+Pl fish(N)+Pl yesterday eat(V)+past+3pc
(Cats ate the fishes yestreday.)

Ex.6.b
avai nalla katal miinkaL.
They(PN) good(Adj) sea(N) fish(N)+Pl
(They are good sea fishes.)

Consider the discourse Ex.6. The plural neuter pronoun 'avai' in Ex.6.b, refers to 'miiNkaL' (fishes) in Ex.6.a. But the plural pronoun resolution engine identifies 'puunaikaL' (cats) which occur as the first NP in the sentence. Plural pronouns such as 'naangkaL' (we), 'engkaL' (our) occur in discourse with explicit antecedent in the discourse. The antecedent has to be understood as the group related to the speaker. These kinds are plural pronouns are not handled.

Noun-Noun anaphora resolution engine fails to handle definite NPs, as in Tamil we do not have definiteness marker, these NPs occur as common noun. Consider the following discourse.

Ex.7.a.
maaNavarkaL pooRattam katarkaraiyil
Student(N)+Pl demonstration(N) beach(N)+Loc
nataththinar.
do(V)+past+3pc
(The students did demonstartions in the beach.)

Ex.7.b.
kavalarkaL maaNavarkaLai kalainthu_cella
Police(N)+Pl students(N) disperse(V)+INF
ceythanar.
do(V)+past+3pc
(The police made the students to disperse.)

Consider the discourse Ex.7. Here in both the sentences 'maaNavarkaL' (students) has occurred referring to the same entity. But these plural NPs occur as a common nons and the definiteness is not signalled with any markers. So we have not handled these kinds of definite NPs which occur as common nouns.

5 Conclusion

We have presented a methodology to build co-reference chains in Tamil text. Co-reference chains are formed by the co-referential entities, which bring cohesiveness to the text. Co-referential entities include pronominals, pronouns with split-antecedents, reflexives, reciprocals, distributives, noun-noun anaphora and definite descriptions and their antecedents. Each of the anaphoric expressions is resolved using different methodologies as pronominal resolution requires syntactic features and noun-noun anaphora resolution and definite description identification requires semantic features. The co-reference chains are evaluated with standard metrics namely MUC, B-Cubed, CEAFe, CEAFm, and BLANC. The average precision is 53.37%, recall of 47.19% and F-measure of 48.79%.

Reference

Ananth Ramakrishnan, A & Sobha Lalitha Devi 2010, 'An alternate approach towards meaningful lyric generation in Tamil', Proceedings of the Workshop on Computational Approaches to Linguistic Creativity (CALC 2010), Association for Computational Lingusitics (ACL), LA, USA, pp. 31-39.

Aone, C & Bennett, S 1995, 'Evaluating automated and manual acquisition of anaphora resolution strategies'. In: 33rd Annual Meeting of the Association for Computational Linguistics, pp. 122-129.

Bengtson, E & Roth, D 2008, 'Understanding the value of features for coreference resolution', In Proceedings of EMNLP, pp. 294-303.

Cardie, Claire & Kiri Wagstaff 1999, 'Noun phrase coreference as clustering', In Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, pp. 82-89.

Culotta, A, Wick, M, Hall, R & McCallum, A 2007, 'First-order probabilistic models for coreference resolution', In Proceedings of HLT/NAACL, pp. 81-88.

Kehler Andrew 1997, 'Probabilistic coreference in information extraction', In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing, pp. 163-173.

Kudo Taku. 2005. *CRF++, An Open Source Toolkit for CRF* [online] http://crfpp.sourceforge.net (accessed 3 January 2013).

Malarkodi C. S., Pattabhi R. K. Rao and Sobha Lalitha Devi. 2012, Tamil NER – Coping with Real Time Challenges, In Proceedings of Workshop on Machine Translation and Parsing in Indian Languages, COLING 2012, Mumbai, India

McCarthy, JF & Lehnert, WG 1995, 'Using decision trees for coreference resolution', In C. Mellish (Ed.), Fourteenth International Conference on Artificial Intelligence, pp. 1050-1055.

MUC-6 1995, Coreference task definition (v2.3, 8 Sep 95). In Proceedings of the Sixth Message Understanding Conference (MUC-6), pp. 335-344.

MUC-7 1997, Coreference task definition (v3.0, 13 Jul 97). In Proceedings of the Seventh Message Understanding Conference (MUC-7).

Ng, V & Cardie, C 2002, 'Improving machine learning approaches to coreference resolution', In. 40th Annual Meeting of the Association for Computational Linguistics, pp. 104-111.

Rahman, A & Ng, V 2011, 'Narrowing the Modeling Gap: A Cluster-Ranking Approach to Coreference Resolution', Journal of Artificial Intelligence Research, vol. 40, pp. 469-521 R.

Raghunathan, K, Lee, H, Rangarajan, S, Chambers, N, Surdeanu, M, Jurafsky, D & Manning, C 2010, 'A multi-pass sieve for coreference resolution', In Proceedings of EMNLP, pp. 492-501.

Ram, RVS, Bakiyavathi, T, Sindhujagopalan, R, Amudha, K & Sobha, L., 2012, 'Tamil Clause Boundary Identification: Annotation and Evaluation', In the Proceedings of 1st Workshop on Indian Language Data: Resources and Evaluation, Istanbul

Ram, RVS & Sobha Lalitha Devi 2016, 'How to Handle Split Antecedents in Tamil?', In proceedings of Coreference Resolution Beyond OntoNotes co-located with NAACL 2016, San Diego, California.

Recasens, M, M`arquez, L, Sapena, E, Mart´I, MA, Taul´e, M, Hoste, V, Poesio, M & Versley, Y 2010, 'SemEval-2010 Task 1: Coreference Resolution in Multiple Languages', In Proceedings of the 5th International Workshop on Semantic Evaluation, ACL 2010, Uppsala, Sweden, pp. 1-8.

Sobha, L 2007, 'Resolution of Pronominals in Tamil. Computing Theory and Application', The IEEE Computer Society Press, Los Alamitos, CA, pp. 475-79.

Sobha Lalitha Devi, Marimuthu K, Vijay Sundar Ram R, Bakiyavathi T and Amudha K. 2013, Morpheme Extraction in Tamil using Finite State Machines, In:Proceedings of Morpheme Extraction Task at FIRE 2013

Sobha Lalitha Devi, Pattabhi RK Rao and R Vijay Sundar Ram. 2016b, "AUKBC Tamil Part-of-Speech Tagger (AUKBC-TamilPoSTagger2016v1)". Web Download. Computational Linguistics Research Group, AU-KBC Research Centre, Chennai, India, 2016.

Soon WH Ng & Lim, D 2001, 'A machine learning approach to coreference resolution of noun phrases', Computational Linguistics, vol. 27, no. 4, pp. 521-544.

Cross Linguistic Variations in Discourse Relations among Indian Languages

Sindhuja Gopalan
AU-KBC Research Centre
MIT Campus of Anna University, Chennai, India
sindhujagopalan@au-kbc.org

Lakshmi s
AU-KBC Research Centre
MIT Campus of Anna University, Chennai, India
lakssreedhar@gmail.com

Sobha Lalitha Devi
AU-KBC Research Centre
MIT Campus of Anna University, Chennai, India
sobha@au-kbc.org

Abstract

This paper summarizes our work on analysis of cross linguistic variations in discourse relations for Indo-Aryan language Hindi and Dravidian languages Malayalam and Tamil. In this paper we have also presented an automatic discourse relation identifier, which gave encouraging results. Analysis of the results showed that some complex structural inter-dependencies existed in these three languages. We have described in detail the structural inter-dependencies that occurred. Discourse relations in the three languages thus exhibited complex nature due to the structural inter-dependencies.

1 Introduction

Discourse relations link clauses in text and compose overall text structure. Discourse relations are used in natural language processing (NLP), including text summarization and natural language generation. The analysis and modeling of discourse structure has been an important area of linguistic research and it is necessary for building efficient NLP applications. Hence the automatic detection of discourse relation is also important. The Indo-Aryan (Hindi) and Dravidian languages (Malayalam and Tamil) share certain similarities such as verb final language, free word order and morphologically rich inflections. Due to the influence of Sanskrit in these languages they are similar at lexical level. But structurally they are very different. In this work we have presented an analysis of the cross linguistic variations in the discourse relations among three languages Hindi, Malayalam and Tamil. Instead of identifying all possible discourse relations we have considered the analysis of explicit discourse relations and developed an automatic discourse relation identification system. During error analysis various structural interdependencies were also noted.

Discourse tagging for Indian languages Hindi, Malayalam and Tamil has been done by Sobha et al., (2014) Other published works on discourse relation annotations in Indian languages are in Hindi (Kolachina et al., (2012); Oza et al., (2009)) and Tamil (Rachakonda and Sharma (2011)). Menaka et al., (2011) in their paper have automatically identified the causal relations and have described about the structural interdependencies that exist between the relations. Similarly, we observed the existence of structural interdependencies between the discourse relations in three languages, which we have explained in detail. From the previous works on discourse relation annotation for various Indian languages, we can observe that the study of discourse relations is carried out for specific Indian language and hence we attempted to discuss the cross linguistic variations among Hindi, Tamil and Malayalam languages.

Researchers have performed identification and extraction of discourse relation using cue based or statistical methods. Penn Discourse Tree Bank (PDTB) is the large scale annotated corpora of linguistic phenomena in English (Prasad et al., 2008). The PDTB is the first to follow the lexically grounded approach to annotation of discourse relations. Marcu and Echihabi (2012) have focused on recognition of discourse relation using cue phrases, but not extraction of arguments. Wellner and Pustejovksy (2007) in their study considered the problem of automatically identifying the arguments of discourse connectives in PDTB. They re-casted the problem to that of identifying the argument heads, instead of

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 402–407

identifying the full extents of the arguments as annotated in PDTB. To address the problem of identifying the arguments of discourse connectives they incorporated a variety of lexical and syntactic features in a discrimination log-linear re-ranking model to select the best argument pair from a set of N best argument pairs provided by independent argument models. They obtained 74.2% accuracy using gold standard parser and 64.6% accuracy using automatic parser for both arguments. Elwell and Baldridge (2008) have used models tuned to specific connectives and connective types. Their study showed that using models for specific connectives and types of connectives and interpolating them with a general model improves the performance. The features used to improve performance include the morphological properties of connectives and their arguments, additional syntactic configuration and wider context of preceding and following connectives. The system was developed on PDTB. They used Maximum entropy ranker. Models were trained for arg1 and arg2 selection separately. They achieved 77.8% accuracy for identifying both arguments of connective for gold standard parser and 73.6% accuracy using automatic parser. Ramesh and Yu (2010) have developed a system for identification of discourse connectives in bio-medical domain. They developed the system on BioDRB corpus using CRFs algorithm. For PDTB data they obtained F-score of 84%. They obtained F-score of 69% for BioDRB data. For PDTB based classifier on BioDRB data, they obtained F-score of 55%. In this work they did not focus on identification of arguments. Versley (2010) presented his work on tagging German discourse connectives using a German–English parallel corpus. AlSaif (2012) used machine learning algorithms for automatically identifying explicit discourse connectives and its relations in Arabic language. Wang et al., (2012) used sub-trees as features and identified explicit and implicit connectives and their arguments. Zhou et al., (2012) presented the first effort towards cross lingual identification of the ambiguities of discourse connectives. Faiz et al., (2013) did explicit discourse connectives identification in the PDTB and the Biomedical Discourse Relation Bank (BDRB) by combining certain aspects of the surface level and syntactic feature sets. In this study we tried to develop a discourse parser for all three languages for identification of connectives and its arguments.

Following sections are organized as follows. Corpus Collection and Annotation is described in section 2, cross linguistic variations in discourse relations among three languages is given in section 3, method used for the automatic identification of discourse relation and the results are described in section 4 and the various structural interdependencies that occur in the three languages is described in section 5. The paper ends with the conclusion section.

2 Corpus collection and Annotation

Health related articles were chosen from web and after removing inconsistencies like hyperlinks a total corpus of 5000 sentences were obtained. Then we annotated the corpus for connectives and its arguments. The discourse relation annotation was purely syntactic. The arguments were labeled as arg1 and arg2 and arg2 was chosen to be following arg1. When free words occur, we tag them separately and the discourse unit between which the relation is inferred is marked as arg1 and arg2. When the connectives exist as bound morphemes we keep them along with the word to which it is attached and include it under arg1. The annotated corpus contains 1332 explicit connectives in Hindi, 1853 in Malayalam and 1341 in Tamil. From the data statistics we can observe that Malayalam language has more number of connectives than Tamil and Hindi. Annotated corpus is used to train the system and the models are built for the identification of connectives and arguments.

3 Cross Linguistic variations in Discourse Relations

The discourse relation in Indian language can be expressed in many ways. It can be syntactic (a suffix) or lexical. It can be within a clause, inter-clausal or inter-sentential. The various cross linguistic variations in discourse relation among the three languages is analyzed and described below.

3.1 Discourse Connectives

Discourse relations can be inferred using Explicit or Implicit connectives. Explicit connectives connect two discourse units and trigger discourse relation. The explicit connectives can be realized in any of the following ways.

- Subordinators that connect the main clause with the subordinate or dependent clause. (For example: agar-to, jabkI in Hindi, appoL, -aal in Malayalam and -aal, ataal in Tamil).
- Coordinators which connect two or more items of equal syntactic importance.

They connect two independent clauses. (For example: “aur”, “lekin” in Hindi, “-um”, “ennaal” in Malayalam and “anaal”, “athanaal” in Tamil).

- Conjunct adverbs that connect two independent clauses and modify the clauses or sentences in which they occur. (For example: “isliye”, “halaanki” in Hindi, “athinaal”, “aakayaal” in Malayalam and “enninum”, “aakaiyaal” in Tamil).
- Correlative conjunctions which are paired conjunctions. They link words or group of words of equal weights in a sentence. (For example: “na keval balki” in Hindi, “maathramalla-pakshe” in Malayalam and “mattumalla-aanaal” in Tamil).

3.2 Position of Connectives

In our approach we have done a syntactic based tagging. In Hindi, Malayalam and Tamil discourse connectives can occur within a sentence or between sentences. In all the three languages inter sentence connectives are said to occupy sentence initial position. Example 1 shows the inter sentence discourse relation in Malayalam.

Example 1:

[chila aaLukaL mukhsoundaryam koottaan
Some people facial-beauty increase
kreemukaL upayogikkaaruNt.]/arg1
creams use
ennaal [athu guNathekkaaLeRe doshamaaN
But that goodness-more than harm-is
cheyyuka.]/arg2
do

(Some people use creams to increase their facial beauty. But that will do more harm than good.)

We found that there exists a difference in the position of conjunct adverb “although” among the three languages. As in Example 2, in Hindi this connective occurs in the sentence initial position whereas in Tamil and Malayalam this connective occurs in the middle position and remains agglutinated with the verb.

Example 2:

haalaaMki [yoga pakshaaGaath kii samasyaa kaa
although yoga paralysis problem's
sTaayii samaaDhaan karthaa hai]/arg2,
permanent solution do is
[yah samay lethaa hai evaM shramsaaDya
This time take is and painstaking
hai]/arg1
is

(Although yoga gives a permanent solution for paralysis, this is time taking and painstaking.)

In Tamil and Malayalam the connective “and” exists in the form as in Example 3. In Hindi single lexicon “aur” serves this purpose.

Example 3:

[muuttukaLiluLLa kuRuththelumpu vaLaraamal
in knee cartilage without
theymaanam atainthaal**um**]/arg1,
growing wear if get-and
[angkuLLa vazhuvazhuppaana thiravam
there smooth fluid
kuRainthupoonaal**um**]/arg2 muuttukaLil uraayvu
get less-and knee friction
eRpatum.
will develop

(If cartilage in the knee gets wear without growing and if the smooth fluid present there becomes less, friction will develop in the knee.)

3.3 Agglutinated and intra sentence

In Malayalam and Tamil connectives can occur as free words or bound morphemes. But in Hindi only free word connectives exist as in Example 2.

Example 4:

[vayiRRil kutalpun irunth**aal**]/arg1 [vayiRu
In stomach ulcer is there-if stomach
valikkum]/arg2.
will pain

(If there is ulcer in stomach, stomach will pain.)

3.4 Paired connectives

In Hindi some discourse connectives were seen as paired connectives. This type of connectives is not noticed in Malayalam and Tamil.

Example 5:

yadhii [lagaathaar buKaar aa rahaa hai]/arg1 **tho**
if constantly fever coming is then
[uskii jaaNca avashaya karaaye]/arg2.
its check sure do

(If fever is coming constantly, then check it for sure.)

In the above Example 5 “yadhii-to” is the paired connective that occurs at the start of arg1 and arg2. Whereas in Tamil and Malayalam it occurs as a single connective as in Example 4 and occurs agglutinated with verb.

3.5 Arguments of Relations

In our approach the label assignment is syntactic. Sometimes, the arguments can be in the same sentence as the connective. Sometimes, one of the preceding sentence acts as an argument. Also the argument can be a non-adjacent sentence. But the text span follows the minimality-principle. In Example 1 the connective “ennal” in Malayalam

connects two discourse units inter sententially. The discourse unit that follows the connective is arg2 and the preceding unit is arg1. In Example 4 the arguments for connective "-aal" in Tamil occur in same sentence.

4 Automatic identification of discourse relation

4.1 Method Used

We have used the method adopted by Menaka et al., (2011) for the identification of discourse relations. We have preprocessed the text for morph analysis (Ram et al, 2010), part-of-speech tagging (PoS) (Sobha et al, 2016), chunking (Sobha and Ram, 2006), clause tagging (Ram et al, 2012). The implementation is done based on machine learning technique CRFs.

4.2 Conditional Random Fields

CRFs is an undirected graphical model, where the conditional probabilities of the output are maximized for a given input sequence. We chose CRFs, because it allows linguistic rules or conditions to be incorporated into machine learning algorithm. Here, we have used CRF++ (Kudo, 2005), an open source toolkit for linear chain CRFs.

4.3 Features Used

For the identification of connectives, we have used PoS tagging information, morphological suffixes and clause information as features for Malayalam and Tamil. Morphological suffixes such as conditional markers, causal markers, relative participle (RP) marker followed by postposition (PSP) and coordination markers were used. For connective identification in Hindi, word, PoS tagging information and chunk information were used. For argument identification we have taken PoS tagging information, chunk information, morphological suffixes, and clause information, combination of PoS and chunk information and connectives as features.

4.4 Training and Testing

For identifying the discourse connectives, we trained the system using the features for connectives. In the next stage we train the system to identify the arguments and their text spans. Here we have built 4 language models for each of the 4 boundaries – Arg2-START, Arg1-END, Arg1-START and Arg2-END motivated by the work of Menaka et al., (2011). The system was trained in 4 phases to develop 4 models. We used 4000 sentences from the corpus for training and 1000 sentences for testing. For testing, the sentences are pre-processed similarly as training data. The system identified the discourse markers in stage 1 and this output becomes input to stage 2. In both the stages we used CRFs as the machine learning algorithm.

The performance of our system is measured in terms of Precision, Recall and F score. Precision is the number of discourse relations correctly perceived by the system from the total number of discourse relations identified, Recall is the number of discourse relations correctly detected by the system by the total number of discourse relations contained in the input text and F-score is the harmonic mean of precision and recall.

The results for connective identification are tabulated in Table 1.

	Precision	Recall	F-score
Hindi	96.33	92.3	94.27
Malayalam	96.3	91.6	93.89
Tamil	95.35	94.18	94.76

Table 1: Results for Connective Identification

The argument identification results are given in Table 2, Table 3, Table 4 and Table 5.

	Precision	Recall	F-score
Hindi	76	72.2	74.05
Malayalam	78.5	72	75.1
Tamil	81.53	73.6	77.36

Table 2: Results for ARG1 Start

	Precision	Recall	F-score
Hindi	75.9	72.2	74
Malayalam	78.8	72	75.23
Tamil	82	72.6	77

Table 3: Results for ARG1 End

	Precision	Recall	F-score
Hindi	77.4	73.2	75.24
Malayalam	79.2	73	75.97
Tamil	81.5	72.6	76.79

Table 4: Results for ARG2 Start

	Precision	Recall	F-score
Hindi	76.3	71.2	73.66
Malayalam	78.7	72.4	75.42
Tamil	82	72.7	77

Table 5: Results for ARG2 End

During error analysis it is noted that a good number of errors are due to structural interdependencies between discourse relations. When there are such structures, there is a considerable overlap in the arguments of two discourse relations leading to the improper identification of boundaries by the system. These are discussed in detail in the next section.

5 Structural Interdependencies between discourse relations

Some very unique pattern of interdependencies was seen existing between discourse relations for Hindi, Malayalam and Tamil mainly due to the free word order nature of those languages. Given below are such patterns.

5.1 Embedding within itself

Due to the free word order nature of Indian languages this type of structure comes into being. Consider the Malayalam Example 6 given below.

Example 6:

[pala padhathikaLum [ee karaaR
many plans this contract
sambhavikkaathathinaal]/arg1 natakkaathe
not-happen-hence failed
poyi.]/arg2

(This contract didn't happen, hence many plans failed.)

Here arg1 and marker is seen embedded inside arg2.

5.2 Between Two Discourse Relations – Containment

One most frequently occurring structural dependency is that of embedding or containment of the whole of a discourse relation within one of the arguments of another discourse relation.

Example 7:

[lagbhag 25 se 50 prathishath roobelaa
approximately 25 from 50 percent rubella
saMkramaN kaa pathaa nahiM cal paathaa]/arg1$_i$
infection know not get
aur [**agar[** iske lakshaN paidhaa hothe
and if its symptoms develop
haiM]/arg1$_j$ **tho** [[ve bhahuth hii
is then they very
halke hothe haiN]/arg2$_i$]/arg2$_j$
light is

(Approximately 25 to 50 percent of rubella infection is not known and if its symptoms develop then they are very light.)

The Example 7 shows that the arguments of connective "agar-to" are contained within the arg2 of connective "aur".

5.3 Between two Discourse Relations – Complete Overlap/Shared Argument

An argument may be shared by two discourse relations in different ways.

Example 8:

naviina vaazhkkai muRaiyil vaakanagkalaip
modern life style vehicles
payanpatutthuv**athaal**]/arg1$_i$ [[nataippayiRci
use-because walking
enpathu kuRainthuvittathu]/arg2$_i$]/arg1$_j$.
is reduced
ithanaal [utalil cerum
Because of this in body accumulate
thevaiyaRRa kalorikaL cariyaaka
unwanted calories correctly
erikkappatuvathillai]/arg2$_j$.
not burnt

(Because of using vehicles in modern life style walking is reduced. Because of this, the unwanted calories accumulated in the body is not burnt.)

In Example 8 the arg2 of the first discourse relation is the shared argument for the second discourse relation.

5.4 Completely Independent Relations

Example 9:

[poshakaaharam nalki kuttiye
nourishing-food gave child
paripaalichu.]/arg1$_i$ **engilum** [kuttiyute
fostered But child's
arogyathil purogathiyilla.]/arg2$_i$ [atuthaghathathil
health-in no-progress next-stage-in
guLikakaL nalki.]/arg1$_j$ **engilum** [kuttiyte
vitamin tablets gave But child's
arogyam athe nilayil thutarnnu.]/arg2$_j$
health same condition-in continued.

(Nourishing food was given for the child. But the child's health had no progress. In the next stage gave vitamin tablets. But the child's condition remained the same.)

In Example 9 there are two adjacent discourse relations which are independent of each other.

6 Conclusion

We have presented our work on discourse relation identification for Hindi, Malayalam and Tamil. An analysis of the discourse relations among the three languages was performed and an automatic identification system for discourse relation was developed. By analyzing the results

structural dependencies were noted. By handling this issue the performance of the system can be improved which makes up our future work.

Reference

John L AlSaif. 2012. Human and automatic annotation of discourse relations for Arabic, Ph.D. thesis, University of Leeds.

Ben Wellner and James Pustejovsky. 2007. Automatically Identifying the Arguments of Discourse Connectives, *Proceedings of EMNLP-CoNLL*, Prague, 92-101.

Balaji P. Ramesh and Hong Yu. 2010. Identifying discourse connectives in biomedical text, *Proceedings of AMIA Annual Symposium*, Washington, DC 657-661.

Daniel Marcu and Abdessamad Echihabi. 2012. An unsupervised approach to recognizing discourse relations, *Proceedings of 40th Annual Meeting on Association for Computational Linguistics*, 368-375.

Robert Elwell and Jason Baldridge. 2008. Discourse connective argument identification with connective specific rankers, *Proceedings of International Conference on Semantic Computing*, Santa Clara, CA.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0, *Proceedings of Language Resources and Evaluation Conference*, Marrakech, Morocco.

Lanjun Zhou, Wei Gao, Binyang Li, Zhongyu Wei, and Kam-Fai Wong. 2012. Cross-Lingual Identification of Ambiguous Discourse Connectives for Resource-Poor Language, *Proceedings of International Conference on Computational Linguistics*, Mumbai, India, 1409-1418.

Ram, RVS, Bakiyavathi, T, Sindhujagopalan, R, Amudha, K and Sobha, L. 2012. Tamil Clause Boundary Identification: Annotation and Evaluation, *Proceedings of 1st Workshop on Indian Language Data: Resources and Evaluation*, Istanbul.

Ravi T. Rachakonda, and Dipti M. Sharma. 2011. Creating an annotated Tamil corpus as a discourse resource, *Proceedings of 5th Linguistic Annotation Workshop*, Portland, Oregon, 119-123.

Sudheer Kolachina, Rashmi Prasad, Dipti M. Sharma, and Aravind Joshi. 2012. Evaluation of Discourse Relation Annotation in the Hindi Discourse Relation Bank, *Proceedings of Language Resources and Evaluation Conference*, Istanbul, Turkey, 823-828.

S. Menaka, Pattabhi R.K. Rao, and Sobha L. Devi. 2011. Automatic identification of cause-effect relations in tamil using CRFs, *Proceedings of Computational Linguistics and Intelligent Text Processing, Lecture Notes in Computer Science*, 6608:316-327.

Sayeed I. Faiz, and Robert E. Mercer. 2013. Identifying explicit discourse connectives in text, *Advances in Artificial Intelligence, Lecture Notes in Computer Science*, 7884:64-76.

Sobha, L and Vijay Sundar Ram, R. 2006. Noun Phrase Chunker for Tamil, *Proceedings of the First National Symposium on Modeling and Shallow Parsing of Indian Languages (MSPIL)*, IIT Mumbai, India, 194-198.

Sobha L. Devi, S. Lakshmi, and Sindhuja Gopalan. 2014. Discourse Tagging for Indian Languages, *Proceedings of Computational Linguistics and Intelligent Text Processing*, Berlin, Heidelberg, 469-480.

Sobha L Devi, Pattabhi RK Rao and Vijay Sundar Ram, R. 2016. *AUKBC Tamil Part-of-Speech Tagger (AUKBC-TamilPoSTagger 2016v1)*, web download, http://www.au-kbc.org/nlp/corpusrelease.html.

Taku Kudo. 2005. CRF++, an open source toolkit for CRF, http://crfpp.sourceforge.net.

Umangi Oza, Rashmi Prasad, Sudheer Kolachina, Dipti M. Sharma, and Aravind Joshi. 2009. The Hindi discourse relation bank, *Proceedings of Third Linguistic Annotation Workshop*, 158-161.

Vijay Sundar Ram, R, Menaka, S and Sobha Lalitha Devi. 2010. Tamil Morphological Analyser", in "Morphological Analysers and Generators, LDC-IL, Mysore, 1 –18.

Xun Wang, Suj Ian Li, Jiwei Li, and Wenj Le Li. 2012. Implicit Discourse Relation Recognition by Selecting Typical Training Examples, *Proceedings of International Conference on Computational Linguistics*, Mumbai, India, 2757-2772.

Yannick Versley. 2010. Discovery of ambiguous and unambiguous discourse connectives via annotation projection, *Proceedings of Workshop on Annotation and Exploitation of Parallel Corpora (AEPC)*, 83-82.

RULE BASED APPROCH OF CLAUSE BOUNDARY IDENTIFICATION IN TELUGU

Ganthoti Nagaraju
Department of Linguistics and Language Technology, Central University of Kerala
gnagarajug62@gmail.com

Thennarasu S
Department of Linguistics and Language Technology, Central University of Kerala
thennarasus@gmail.com

Christopher Mala
Center for Applied Linguistics and Translation Studies, University of Hyderabad
efthachris@gmail.com

Abstract

One of the major challenges in Natural Language Processing is identifying Clauses and their Boundaries in Compu-tational Linguistics. This paper attempts to develop an Automatic Clause Bound-ary Identifier (CBI) for Telugu lan-guage. The language Telugu belongs to South-Central Dravidian language fami-ly with features of head-final, left-branching and morphologically agglutinative in nature (*Bh. Krishnamurti*, 2003). A huge amount of corpus is studied to frame the rules for identifying clause boundaries and these rules are trained to a computational algorithm and also discussed some of the issues in identifying clause boundaries. A clause boundary annotated corpus can be developed from raw text which can be used to train a machine learning algorithm which in turns helps in development of a Hybrid Clause Boundary Identification Tool for Telugu. Its implementation and evaluation are discussed in this paper.

1. Introduction

A Clause is a grammatical unit that includes, at minimum, a predicate and an explicit and implied subject and expresses a proposition (*Crystel*, 1980). In other words, a clause is defined as a group of words having a subject and a predicate. It is a well-known fact that a sentence may contain one or more clause. Simple sentences always have a single clause. Analyzing these clauses in NLP is an easy task. But when a sentence has more than one clause it becomes difficult to process. Identification of predicate and its dependent thematic elements become even more difficult. To solve this problem, identification of clause boundary is mandatory. Clause Boundary Identification is the process of dividing the given sentence into a set of clause. Correct automatic detection of major syntactic boundaries, in particular clause boundaries help in improving many other tools in NLP (*Leffa*, 1998; *Ejerhed*, 1988, *Vijay et al*., 2009; *Gadde et al*., 2010). The ***Telugu Clause boundary identifier (T-CBI)*** is an automatic tool to identify boundaries of clauses and mark their start and end points. In another words, it identifies the structure that underlies the sentence. Shallow parsed sentence are used as input to T-CBI and further parse the sentence and marks the clause with their boundaries along with their appropriate tags. This module can be used in bigger NLP systems like Machine Translation systems, Information Extraction and Information Retrieval, Search Engines, etc.

The data driven (*Puscasu*, 2004) and rule based (*Leffa*, 1998) approaches are prominent in the development of a CBI. In order to build a clause boundary identifier, using data driven approach, one needs to have a good clause boundary annotated corpus for training (*Sharma et al*, 2013). Such a corpus is not available in the Telugu language. Hence a Rule-Based approach is selected in the current study to develop an efficient CBI for Telugu. By using Morphological cues such as agreement markers (person, gender, number) and case markers/ Tense Aspect Modal

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 408–416

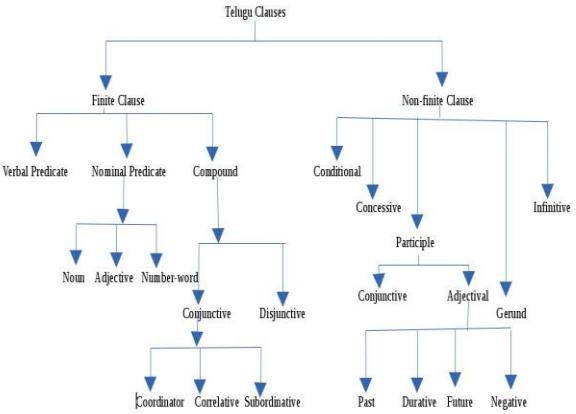
(TAM) markers to identify the start and end of the clause. Other than these, certain lexical cues are used to identify the CBI. Identified Thematic roles of the constituents are used in the T-CBI for better performance of the Rules. The development of automatic T-CBI will be used to develop the clause boundary annotated corpus for the task of clause boundary identification from raw text using machine learning process in NLP.

2. Review of Earlier Researches

Clause Boundary Identification started with *Eva Ejerhed's* Clause Identification System (*Ejerhed*, 1988) for text to speech system. *Leffa* (1998) has developed a rule-based system to identify clauses from English to Portuguese machine translation system. *Papageorgiou* developed a rule-based clause boundary system as a pre-processing tool for bilingual alignment parallel text (*Papageorgiou*, 1997). *Tomohiro Ohno et al.* (2006) built CBI for Japanese to implement Spoken Monologue System. The dependencies within a clause are identified by dividing a sentence into clauses and executing stochastic dependency parsing for each clause. Later, the dependencies over clause boundaries are identified stochastically, and the dependency structure of the entire sentence is thus completed. This method executes dependency parsing in two stages: at the clause level and at the sentence level. According to their evaluation, the recall of the system is 95.7% and the precision is 96.9%. *Phani Gadde et al.* (2010) have attempted to improve data driven dependency parsing using clausal information. They have used Stagel parser of *Husain et al*. (2009), to provide the clause boundary information that is then incorporated as features during the actual parsing process. They experimented with different combinations of the information provided in the data such as Vibhakti and TAM fields. *Daraksha Parveen et al.* (2011) have built a CBI for Urdu using Conditional Random Field (CRF) as the classification method and clause markers. A hybrid approach is proposed to use both techniques i.e. rule based and machine learning to build an identifier for different clause boundaries of Urdu language. *Lakshmi, S. et al.* (2012) have built a clause boundary identification system for *Malayalam* sentences using the CRF. Here, the clause boundaries are identified using grammatical features. *Sobha L. et al.* (2013) have built Malayalam CBI using CRF. They have developed a corpus with tagging of different type of clauses as well as the start and end of the clause. They selected approximately 6415 tourism and 385 health corpus sentences from the Web and the training set consisted of 5000 sentences from both the domains. Testing of the system was done with 401 unseen sentences from the corpus of tourism. They achieved a precision and recall on different types of clauses of about 70% and 80% respectively. *Aniruddha Ghosh et al.* (2013) have built CBI for the Bengali language. They used a syntactic rule based model with CRF, a machine learning technique. They have achieved 73% and 78% of precision and recall respectively. *Rahul Sharma et al.* (2013) have attempted to build a clause boundary to Hindi Treebank data. They have used the dependency attachments and dependency annotated relations to mark clauses. They chose 16,000 sentences and conducted an exercise on 238 clauses and got the result of 94% of accuracy in the clause boundary identification.

3. Description of Rule Format.

Classification of Telugu Clauses



3.1. Finite Clauses

Finite Clause contains subject and finite verb, the finite verb is in agreement with the subject. Finite clauses are classified as Verbal Predicate, Nominal Predicate and Compound. Nominal predicate in sub-categorized to Noun, Adjectival and Number-words and Compound Clauses are sub-categorized to Conjunctive and Disjunctive Clauses. Conjunctive clauses are further sub-categorized into co-ordinate, correlate and subordinate clauses.

3.1.1. Verbal Predicate clause

In verbal predicate clause, a finite verb occurs in the end of a sentence. It distinguishes for tense such as past, present and future. Each tense distinguishes positive and negative clauses. Examples for verbal predicate clauses are given below:

Past Tense

i) Positive

nēnu āme-nu tiṭṭ ā-nu
I she ACC scold PST 1.SG.
'I scolded her'

ii). Negative

nēnu āme-nu tiṭṭ a-lēdu
I she ACC scold INF- PST. NEG
'I did not scold her'

3.1.2. Nominal Predicate clause

Noun/adjective/number word phrase in its predicate position agrees in gender, person and number with the subject. In the case of first or second person singular or in first person plural in Telugu, they inflect with number and with person. The pronominal suffixes -ni/-nu, -mi/-mu and -vi/-vu are manifested on the predicate for first person singular, first person plural and second person singular respectively.

i). Noun phrase in Predicate position:

nēnu oka abbāyi- ni
I a boy 1.SG.
'I am a boy'

This is an example for nominal predicate sentence. The subject *nēnu* agrees with the nominal predicate. The agreement is manifested with the marker *-ni*

ii). Adjective phrase in Predicate position:

nīvu/nuvvu poḍugu vāḍi-vi
you tall poss. 2.SG.
'You are a tall boy/guy'

The subject *nīvu/nuvvu* agrees with the adjectival predicate. The agreement is manifested with the marker *-vi*.

iii). Number word phrase in Predicate position:

mēmu muggu- ra-mu
we three 3.SG.H. 1.PL.
'We are three persons'

This is an example for number word predicate sentence. The subject *mēmu* agrees with the number-word predicate. The agreement is manifested with the marker *-mu.* No Nominal Predicate Agreement for 2.PL and 3.SG/PL.M/F/N

3.1.3 Compound Clause

A sentence containing two or more coordinate independent clauses, usually joined by one or more conjunction markers, but no dependent clause, as in the below sentence.

e.g.: [The lightning flashed]$_{MC}$ and [the rain fell.]$_{MC}$

3.1.3a Conjunctive Clause

Some compound sentences are joined by a conjunction. Some of the conjunction marker in Telugu are: ani, mariyu, leka, (k*ā*ni), k*ā*ka, Ena, kanuka, endukante.

Examples of compounds in sentences include:
m*ā* *ā*yana panilo nimaGFulE unn*ā*ru aMdukani nenu baj*ā*ruku velYl*ā*nu.

Other compound sentences are joined with a semicolon. If a semicolon is used, it may or may not have a conjunctive adverb.

i). Coordinative Clause

This is done using one of two or more clauses of equal status in a sentence, especially when joined by a coordinating conjunction. Here *kāni* is the coordinative marker.

mēmu vacc-ā- -mu kāni [vāḍu rā-lēdu
we come PST. 1.PL. but he come.INF PST.NEG
'We came but he did not come.'

ii) Correlative Clause

A correlative conjunction is a paired conjunction that links balanced words, phrases, and clauses. The elements connected by correlative conjunctions are usually parallel - that is, similar in length and grammatical form. Each element is called a *conjoin*. Some of the Correlatives in Telugu are: *leka/kāka/gāka, kādu/kani and "-e kāka/-e gāka/-V kūda.*"

evaḍu vaccāḍ-ō vāḍu nā tammu-ḍu
who come RTM he ACC brother 3.SG.
'The one who came is my brother'

Here we are using morphological cue
X-*ō Y*
ō-relative-correlative marker, X and Y are clauses.

iii) Subordinate Clause

A subordinating conjunction is a word that connects a main clause to a subordinate clause. A main clause is an independent clause that can stand alone by itself as a sentence. In other words, a main clause does not need any additional information to operate as a sentence.
Here we use lexical cues:
X- COMP *Y*
COMP -complementizer, X and Y are clauses.

ani	*annā*
anēdi	*anna*
atlu	*aMte*
annatlu	

MC1[nēnu vacc- ā- -nu]MC1 ani MC2[vāḍi-to cepp- ā- nu]MC2
I come PST 1SG. COMP he ASS. say PST. 1.SG
'I told him that I came'

3.1.3b. Disjunctive Clause

A coordinate construction is the one that uses a *disjunctive conjunction* to indicate a contrast. The items on either side of the disjunctive conjunction are called disjunct.
Here we use morphological cues.
X- *ō Y-ō Z*
ō -Disjunctive marker, X, Y and Z are clauses

vast- ā- ḍ- ō rā- ḍ- ō nāku tēliyadu
come FUT. 3.SG.M DISJ come. FUT. NEG-3.SG.M DISJ I-DAT do not know
`I do not know either he comes or not'

3.2. Non-finite Clauses

Non-finite clause are formed with non-finite verb and verb does not marked with gender, number and person suffixes in agreement with grammatical subject of the sentence, but they form by adding appropriate tense-mode suffix to a verb stem. And they are always depended and embedded.

3.2.1. Conditional clause

In this conditional clause we use morphological cue -te `positive conditional marker'
-rākapōte/ rākuMṭe `negative conditional marker'
SC[nuvvu addamu nu jāraviḍis-tē]SC MC[adi pagilipō- tuM- di]MC
you mirror ACC. drop COND that break FUT 3.SG.N.
'If you drop the mirror, it will break'

This is an example for complex sentence with conditional clause. As we noted here, there are two clauses, one of them is subordinate clause or dependent clause and the other one is super ordinate clause or main clause. The verb is present in both the sentences. The subject *nuvvu* in SC ends with conditional marker -*t*ē and *adi* in MC agrees the agreement and is manifested with the marker -*di* respectively

3.2.2. Concessive clause

In this concessive clause we use morphological cues. *-inā* `positive concessive marker' *-rākapōyinā/ -rākunnā* `negative concessive marker

SC[jōhn veḷl- inā]SC MC[mary veḷla-du]MC
John go CONCM Mary go 3.SG.
'Though John goes, Mary don't go'

This is an example for complex sentence with concessive clause. As we noted here, there are two clauses, one of them is subordinate clause or dependent clause and the other one is super ordinate clause or main clause. The verb is present in both the sentences. The subject *Jōhn* in SC ends with concessive marker *-inā* and *Mary* in MC agrees the agreement and is manifested with the marker *-du* respectively.

3.2.3. Participle Clause

3.2.3a. Conjunctive Participle

We use morphological cues in this conjunctive participle clause. *-i* `positive conjunctive participle marker' *-aka/ akuMḍā* `negative conjunctive participle marker'

MC[SC[ataḍu iḍli tin- i]SC kāṅī coffee trāga lēdu]MC
he idli eat CP coffee drink NEG
'Having eaten idli, he didn't take coffee'

This is an example for complex sentence with verbal participle clause and also forward control. As we noted here, there are to clauses, one of them is subordinate clause or dependent clause and the other one is super ordinate clause or main clause. The verb is present in both the sentences. The subject *ataḍu* in SC ends with verbal participle marker-*i* and in MC agrees the agreement and is manifested with the negative marker *lēdu* respectively.

sujāta snānaM ceyy- aka vāraM rōjulay- iM-di
sujatha bath PV NCP week days PST 3.SG.
'Sujatha has not bathed from the last one week'

According to *Chekuri ramarao,* in his book *Telugu Vakyam,* he explained in this sentence (48) also that it is a temporal expression, even though *'vāraM rōjulayiMdi'* is plural marker it takes only singular agreement marker *-di.* In this case we should not use yesterday, day before yesterday, etc. for this sentence also. Here we have negative participle marker *-aka* is there in the sentence.

3.2.3b. Adjective Participle

We use morphological cues in this adjectival predicate. *-ina* `positive past adjectival participle marker' *-tunna* `positive durative adjectival participle marker' *-ē* `positive future adjectival participle marker' *-ani* `Negative adjectival participle marker'

Adjectival participle form of verbs can be pronominalized.

i) Past Adjectival Participle:

SC[nēnu vāḍi- ki icc- ina pustakaM]SC
I he-DAT give PP book
'I gave a book to him'

This is an example for accusative nominalization in complex sentences with adjectival participle clause. As we noted here, the subject *nēnu* in SC ends with past participle marker-*ina* and the accusative *pustakaM* is nominalized as shown in the example. The actual sentence is *'nēnu vāḍiki pustakaM iccānu',* before doing nominalization *pustakaM* is in the accusative position.

nēnu vāḍi- ki ivv- ani pustakaM
I he DAT give NPP book
'I didn't give a book to him'

This is an example for accusative nominalization in complex sentences with adjectival participle clause. As we noted here, the subject *nēnu* in SC ends with negative past participle marker *-ani* and the accusative *pustakaM* is nominalized as shown in the example. The actual sentence is *' nēnu vāḍiki pustakaM ivva lēdu',* before doing nominalization *pustakaM* is in the accusative position

ii) Durative Adjectival Participle

vāḍi- ki pustakaM is- tunna nēnu
he DAT book give DP i
'I am giving a book to him'

This is an example for nominative nominalization in complex sentences with adjectival participle clause. As we noted here, the subject *vāḍu* in SC ends with durative participle marker -*tunna* and the nominative *nēnu* is nominalized as shown in the example. The actual sentence is *'nēnu vāḍiki pustakaM istunnānu',* before doing nominalization *nēnu* is in the nominative position.

vāḍi- ki pustakaM ivv- ani nēnu
*h*e DAT book give NDP i
'I am not giving a book to him'

This is an example for nominative nominalization in complex sentences with adjectival participle clause. As we noted here, the subject *vāḍu* in SC ends with negative durative participle marker -*ani* and the nominative *nēnu* is nominalized as shown in the example. The actual sentence is *'nēnu vāḍiki pustakaM ivvaḍaM lēdu',* before doing nominalization *nēnu* is in the nominative position

iii) Future Adjectival Participle:

vāḍi- ki pustakaM icc- ē nēnu
he DAT book give FP I
'I will give a book to him'

This is an example for nominative nominalization in complex sentences with adjectival participle clause. As we noted here, the subject *vāḍu* in SC ends with future participle marker -*ē* and the nominative *nēnu* is nominalized. The actual sentence is *'nēnu vAdiki pustakaM iswānu',* before doing nominalization *nēnu* is in the nominative position.

vāḍi- ki pustakaM ivv-ani nēnu
he DAT book give NFP i
'I will give a book to him'

This is an example for nominative nominalization in complex sentences with adjectival participle clause. As we noted here, the subject *vāḍu* in SC ends with negative future participle marker -*ani* and the nominative *nēnu* is nominalized as shown in the example. The actual sentence is *'nēnu vAdiki pustakaM ivvanu',* before doing nominalization *nēnu* is in the nominative position.

3.2.4. Gerundival Clause

In gerundival clause also we use morpholgical cues: -*aḍaM* `gerundival marker'

SC[vāḍi- ki cepp- aḍaM]SC MC[nā- ku isṭaM lēdu]MC
he DAT book GEND me DAT like NEG
'I don't like telling him'.

This is an example for gerundival clause. As we noted here, the subject *vāḍu* in SC ends with gerundival marker -*aḍaM.*

3.2.5. Infinitival Clause

In infinitival clause we use morphological cues. -*a* `infinitive marker', -*a_ gānē* 'as soon as', -*a_baṭṭi* 'because'

āme cepp- a- baṭṭi nēnu cēs- ā- nu
she say INF because I do PST 1.SG
'Because she told, I did'.

This is an example for infinitival clause. As we noted here, the subject *āmē* in SC ends with infinitival marker -*baṭṭi.*

Rule for Clause Identification

Rules for clause boundary identification are described in the below:

Clause type	Clause start	Clause end	Information		
			Chunk	Morph-tam	2nd column
Conditional clause		'-te'(past)	VGNF	'-te'(+ve) '-akapōte'(-ve) '-akuMte'(-ve)	
		'-tunMte'(durative)	VGNF	'-tunMte'(+ve) '-akuMte'(-ve)	
Concessive clause		'-inā' (past)	VGNF	'-inā'(+ve) '-kapōyinā'(-ve) '-akunnā-(-ve)	
		'-tunnā'(durative)	VGNF	'-tunnā'(+ve) '-akunnā-(-ve)	
Verbal participle		'-i'(past)	VGNF	'-i'(+ve) '-aka'(-ve)	
		'-tū'(durative)	VGNF	'-tū'(+ve) '-akuM?ā(-ve)	
Adjectival participle					
1.past		'-ina'	VGNF	'-ina'(+ve) '-ani'(-ve)	
2.durative		'-(t)unna'	VGNF	'(t)unna'(+ve) '-ani'(-ve)	
3.future		'-ē'	VGNF	'-ē'(+ve) '-ani'(-ve)	
Gerundival cluse					
		'-a?aM_ki' '-a?aM_valana/valla'	VGNF VGNF	'-a?aM_ki' '-a?aM_valana/valla'	
		'-a?aM_ce'	VGNF	'-a?aM_ce'	
		'-a?aM_to'	VGNF	'-a?aM_to'	
		'-a?aM_e'	VGNF	'-a?aM_e'	
		'-a?aM_gani'	VGNF	'-a?aM_gani'	
		'-a?aM_0_e' '-a?aM_to_e'	VGNF VGNF	'-a?aM_0_e' '-a?aM_to_e'	
		'-a?aM_nu'	VGNF	'-a?aM_nu'	
		'a?aM_gala_a'	VGNF	'a?aM_gala_a'	
		'a?aM_kūdā'	VGNF	'a?aM_kūdā'	
		'-a?aM_ki'	VGNF	'-a?aM_ki'	
		'-a?aM_kosaM_ainā'	VGNF	'-a?aM_kosaM_ainā'	
Tense		'-ā'(Past)	VGNF	'-ā'(+ve) 'a_lēdu'(-ve)	
		'-tunn'(durative)	VGNF	'-tunn'(+ve) '-a?aM_lēdu'(-ve)	
		'-tā'(Future)	VGNF	'-tā'(+ve) '-a'(-ve)	
Mood / Mode		'-0'(Imperative)	VGNF	'-0'(+ve)	
		'-a_vaddu'(Prohibitivee)	VGNF	'-a_vaddu'(+ve)	
		'-dā'(Hortative)	VGNF	'-āmu'(+ve)	
		'-āli'(Obligative)	VGNF	'-āli'(+ve)	
		'-gala'(Capabilitative)	VGNF	'-gala'(+ve) a_lenu	
Non-finitive		'-an'	VGNF	'-an'	
Auxiliaries					
Negative past		'a_le'	VGNF	'a_le'	
Passive		'-a_ba?u'	VGNF	'-a_ba?u'	
Inceptive		'-a_bo'	VGNF	'-a_bo'	
Inchoative		'-a_nāBiMcu'	VGNF	'-a_nāBiMcu'	
Optative		'-a_gōru'	VGNF	'-a_gōru'	
Permissive		'-a_ivvu'	VGNF	'-a_ivvu'	
Causative		'-a_manu' '-0_komanu'	VGNF	'-a_manu' '-0_komanu'	
Abilitative		'-a_tagu'	VGNF	'-a_tagu'	
Probability		'-a_vaccu'	VGNF	'-a_vaccu'	
Forcive		'-a_valayu' '-a_budDavvu '	VGNF	'-a_valayu' '-a_budDavvu '	
Negative permissive		'-a_kū?adu'	VGNF	'-a_kū?adu'	
Reflective		'-ko'	VGNF	'-ko'	
Reciprocal		'-ko'	VGNF	'-ko'	
Complitive		'-i_pō'	VGNF	'-i_pō'	
Discourse markers:					
And			CCP		'mariyu'
But			CCP/BLK		'kāni'
Or			CCP		'lēka'
Because			CCP		'eMdukanaga/eMdukaMte
So			CCP		'aMduvalla' aMduvalana
If					'okavē?a'
Then			VGNF/CCP		'ayitē'
Even though					'ayinappatiki'
Yet			NP		'iMkā'
Because of that					'dānivalana'
Perhaps			VM		'bahuś ā'
Not only this					'idikā'kuM?ā'
Also			BLK		'kū?a'
Lastly					'civaragā'
Next			NP		'taruvāta'
Meanwhile					'madyalo'

4. Testing and Evaluation

We have taken 5000 sentence from tourism and health domain corpus to test the rule set. This corpus includes all type of sentence like Relative Participle, Conditional, Main clauses and etc.

S. No.	Clause Type	No. of Sent
1	Conditional	907
2	Concessive	280
3	Verbal participle	1017
4	Adjectival participle	674
5	Gerundival	436
6	Non-finite	869
7	Relative	817
	Total	5000

These sentences were first processed with shallow parser from which we extract Morphological, POS and Chunk information later this is given as input to simple parser which assigns the karaka roles (thematic relationships) which indeed helps the T-CBI perform well. Evaluation of the Rules is given below.

Clause Type	No. of Sentence	Correct Identifcation	% of recognition
Conditional	907	857	
Concessive	280	184	
Verbal participle	1017	802	
Adjectival participle	674	490	
Gerundival	436	421	
Non-finite	869	532	
Relative	817	593	
Total	5000	3879	77.58

Out of 5000 sentence T-CBI was able to identify 3879 sentence correctly which means the performance of the T-CBI can be scaled to 77.58% of accuracy. Using this we can create huge amount of corpus of CBI pre-annotated.

5. Conclusion

Thus, this paper has attempted to show how implicit clausal information captured in a shallow parsed text can be extracted and are used to develop CBI for Telugu. The paper has also discussed some of the issues in identifying clause boundaries using the above said approach. A plan has been implemented with these rules in a computational algorithm for future exercises. Once the rules are implemented it would be easy to scale the performance of the rules designed. A clause boundary annotated corpus can be developed from raw text which can be used to train a machine learning algorithm which in turns helps in development of a Hybrid Clause Boundary Identification Tool for Telugu.

Reference

Bharati, S. Husain, B. Ambati, S. Jain, D. Sharma and R. Sangal. 2008. `Two Semantic features make all the difference in Parsing accuracy'. Proc. of ICON-08.

Bharati, Vineet Chaitanya, Rajeev Sangal. *Natural Language Processing A Paninian Perspective.* Prentice Hall of India (1995).

Ghosh, A. Das, and S. Bandyopadhyay, “Clause Identification and Classification in Bengali,” in *Proceedings of the 1st Workshop on South and Southeast Asian Natural language Processing* (WSSANLP, 23rd International Conference on Computational Linguistics (COLING), Beijing, August 2010, pp. 17-25.

E. Ejerhed, “Finding Clauses in Unrestricted Text by Finitary and Stochastic Methods,” in *Proceedings of the 2nd Conference on Applied Natural Language Processing*, Austin Texas, 1988, pp. 219-227.

Gadde, Phani, et al. 2010. "Improving data driven dependency parsing using clausal information." *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Lakshmi S, Vijay Sundar Ram R and Sobha Lalitha Dev. 2012. Clause

Boundary Identification for Malayalam Using CRF. In *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages* (MTPIL).24th International Conference on Computational Linguistic

Sharma, Rahul, et al. 2013. "*Automatic Clause Boundary Annotation in the Hindi Treebank.*" WSSANLP-2013: 83.

"*A rule based approach for automatic Claus boundary detection and classification in Hindi"* by Rahul Sharma and Soma Paul [Proceedings of the Conference the 5th Workshop on South and Southeast Asian NLP WSSANLP - 2014].

Keynote Lecture-3

Towards Abstractive Summarization

Vasudev Verma
International Institute of Information Technology, Hyderabad, India

In this talk, I will be sharing our research journey of building summarization engines to produce various flavors of summaries. Starting from single document summarization, our experience of building multi-document summarization (MDS), query focused MDS, Update or Progressive summarization, guided summarization, comparison summarization and personalized summarization systems can be seen as a movement from Extraction based to abstraction based summary generation. We have used variations of Relevance Based Language Model (RBLM) along with external knowledge sources, dependency parsing and more recently deep learning techniques in building these systems. Since 2006, our team has consistently performed well and ranked as a top team in various tracks of DUC (Document Understanding Conference) and TAC (Text Analysis Conference) conferences conducted by NIST. I will also discuss our recent attempts to create semi-abstractive summarization models using natural language processing based approaches as well as deep learning based approaches.

"Who Mentions Whom?"- Understanding the Psycho-Sociological Aspects of Twitter Mention Network

R.Sudhesh Solomon **Srinivas P Y K L** **Abhay Narayan** **Amitava Das**
Indian Institute of Information Technology, Sri City
{sudheshsolomon.r,srinivas.p,abhay.n,amitava.das}@iiits.in

Abstract

Users in social network either unicast or broadcast their messages. At mention is the popular way of unicasting for Twitter whereas, general tweeting could be considered as broadcasting method. Understanding the information flow and dynamics within a Social Network and modeling the same is a promising and an open research area called Information Diffusion. This paper seeks an answer to a fundamental question - *whether the at-mention or the uni-casting pattern in social media is purely random in nature or there is any user specific selectional preference*? To answer the question we present an empirical analysis to understand the psycho-sociological aspects of Twitter mentions network within a social network community. To understand the psychological pattern we have analyzed personality (Big5 model: *Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism*) of users and to understand the the sociological behavior we analyze values (Schwartz model:*Achievement, Benevolence, Conformity, Hedonism, Power, Security, Self-Direction, Stimulation, Traditional, and Universalism*) of all the users inside a community. Empirical results suggest that personality and values traits are indeed salient cues to understand how the mention-based communication network functions. For example, we notice that achievement-oriented communities talk to each other more often than other people. We also observe that neurotic people are more involved in communication within their community.

1 Introduction

Information diffusion is a process of spreading information or content within a network via a particular path or pattern. A significant amount of research has been done in this area in the past few years. However, most of the previous efforts considered only network topology for the diffusion process.

To understand the propagation process we need to understand who is connected with whom and in what manner. At mention on Twitter is the way of one-to-one conversation. The question we raise here is whether the at-mention pattern is purely random in nature or is there any user specific selectional preference? Selectional preference implies to the choice that certain kind of people make for direct communication but, while they are interested in broadcasting they behave differently. To understand the notion this paper presents an empirical analysis to understand the psycho-sociological aspects of Twitter mentions network within a social network community. To this end, we analyze personality and values of all the users in a social network community. First, we categorize social network communities based on values types and analyze mention network within each type (i.e. power, hedonic, etc.) of communities. We notice that achievement-type communities talk to each other more often than other people. Then we analyze how people with certain personality type (i.e. open, extrovert, etc.) interact with other types of people inside a community. We observe that conscientious people are more involved in communication within their network.

Empirical results suggest that personality and values traits are indeed a salient cue to understand how the mention-based communication network functions. In our analysis, we found that the members from stimulation, achievement, and benevolent oriented communities are closely con-

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 418–426

nected among themselves while the members in other communities do not show significant connectivity among themselves. Thus in our analysis, we were able to find that universal, extrovert, and open people prefer broadcasting over unicasting messages (via @ mention).

Communication dynamics in human society is a complex phenomenon. Here, in this paper, we present empirical results to establish correlations of the user's unicasting behavior vs his/her psycho-sociological traits. We believe that there are many properties that are not considered (content of the message, age, gender) which affect the at-mention dynamics in the social network. However, we are not considering those aspects of this paper. Our future research is motivated towards that direction.

2 Related Work

The research paradigm called information diffusion seeks to answer how information spreads in a social network and model how a given piece of information will propagate through a social network - more precisely what a user will do with a particular tweet (lets say), will he/she either retweet it, atmention somebody or broadcast it again to spread it over to a wider audience within his/her reachability in the network. Essentially researchers seek to answer to the following questions :(i) *which pieces of information or topics are popular and diffuse the most*, (ii) *how, why and through which paths is the information diffusing, and will be diffused in the future*, (iii) *which members of the network play important roles in the spreading process?*

A considerable amount of work has been done in modeling the process of information diffusion in online social networks. Previous works on information diffusion have considered several influencing factors such as speed, scale, range, influential nodes, network topology, topics and etc. In the following paragraphs we are describing such related works.

Research endeavors by (Kimura et al., 2010) and (Wani and Ahmad, 2014) discussed diffusion process based on network topology and they explain about the concept of influential nodes or in simple terms, which node/s will influence the other nodes in the diffusion process. (Kimura et al., 2010) explains about combinatorial optimization problem, which is a way to find out the most influential nodes in a social network. In (Wani and Ahmad, 2014) the authors explain about understanding the dynamics of social networks and modeling the same, dynamics here refer to the topological structure of the network. The authors also explained about various information diffusion parameters (diffusion rate, who influenced whom etc.) in this work. Research by (Gomez Rodriguez et al., 2013), tried to capture time dimension of the diffusion pattern. The main motivation of the authors in this work was to infer the edges and the dynamics of the underlying network.

Some of the other works discussed about the topic based diffusion pattern. Work by (Romero et al., 2011), analyzed diffusion pattern based on hashtags categorizations such as celebrity, games, idioms, movies, tv, music, politics, sports, and technology. To describe the diffusion patterns the authors took two measures - Stickiness: *The measure of the contingency of an information. The peak value of the curve*. Persistence: *The time for which an information stays on a particular diffusion rate. The measure of rate of decay after the peak*. Then they empirically show how topical variations affect stickiness and persistence of information diffusion patterns. The other interesting work by (Apolloni et al., 2009) proposed a probabilistic model to understand how two people will converse about a particular topic based on their similarity: *based on demographic information. The popular idea of homophily and heterophily* and familiarity: *based on time that they spend together in same topic*.

Retweeting is the famous way of information cascading in Twitter. There are research endeavors to predict how retweeting diffusion pattern will be. The work by (Zaman et al., 2010) moduled the information diffusion task as a predicting modeling. Using a large scale data on who has retweeted and what was retweeted a probabilistic collaborative filtering model was built to predict the future retweeting pattern. The model learnt on parameters like the tweet source (the tweeter), the user who was retweeting and the retweet content. Works by (Yang and Counts, 2010) discussed about several influencing factors such as speed, scale and range of retweeting behavior. The first factor analyzed was Speed – *whether and when the first diffusion instance will take place*. To perform the analysis on speed, two models were used. The first model answers when a tweet containing a par-

ticular topic is likely to be mentioned by another tweet containing the same topic. For example, when user A posts a tweet related to a topic XYZ, how quickly another user (say user B), responds to the tweet consisting XYZ mentioning user A. Secondly, the Cox proportional hazards model (Cox and Oakes, 1984) was used to quantify the degree to which a number of features of both users and tweets themselves to predict the speed of diffusion to the first degree offspring. The second factor explained and analyzed in this work is Scale – *the number of affected instances at the first degree*. In this work, the number of times a person is mentioned in the retweet trail relating to a topic was analyzed and a probabilistic diffusion model has been proposed. The last factor considered in this work is Range – *how far the diffusion chain can continue on in depth.* The analysis on range was done by tracing a topic from a given start node to its second and third degree of offspring nodes, and so on.

A few works have discussed about behavior of group of individuals - **Herd Behavior**: a social behavior occurring when a group of individuals make an identical action, not necessarily ignoring their private information signals. However, user level sentimental preference is being ignored so far. Therefore, our current work is on understanding user societal sentiment behavior. Our theoretical point of departure is in psycho-socio-linguistic models, the Schwartz model *Achievement, Benevolence, Conformity, Hedonism, Power, Security, Self-Direction, Stimulation, Traditional and Universalism.*. We hypothesis that people have natural preferences for direct communications. That means certain type of people who possess one value type have preference over other kind of people of different value within their range. For example, we observe that the traditional people are less likely involved in communication (i.e, unicasting) compared to other communities of people of different value types.

3 Computational Psycho-Sociological Models

In recent years, there have been significant efforts on determining the opinion or sentiment or emotion about a specific topic held by the author of a piece of text, and on automatic sentiment strength analysis of text, classifying it into either one of the classes – positive, negative or neutral, or into Ekman's classes – happy, sad, anger, fear, surprise, and disgust. However, grouping people based on positive, negative, or neutral comments and then understanding their behavior would be spurious. Therefore we propose, psycholinguistic and sociolinguistic models in order to capture user's intrinsic selectional preferences.

The Big 5 personality (Goldberg, 1990) model and Schwartz values (Schwartz, 2012) model can be considered as a person level sentiment model (depicted in table 1) and a societal sentiment model, respectively. Traditional sentiment analysis systems detect sentiment at text-level, whereas the personality model aims at understanding the sentiment/personality of each individual whereas the Schwartz model describes the societal sentiment of groups of individuals forming communities in social networks.

Personality- User Level Sentiment: There has been a growing interest in the scientific community on doing automatic personality recognition from their language usage and behaviour in social media. A milestone in this area was the 2013 workshop and shared task on Computational Personality Recognition (WCPR) (Celli et al., 2013), repeated in 2014 (Celli et al., 2014). Two corpora were released for the 2013 task. One is a Facebook corpus, consisting of about 10,000 Facebook status updates from 250 users, plus their Facebook network properties, labelled with personality traits. The other corpus comprises 2,400 essays written by several participants labelled with personality traits. The best performing system (F-score = 0.73) was developed by (Verhoeven et al., 2013). The various features and methods used by all the participant groups can be viewed as either linguistic and non-linguistic. Another relevant research work on developing computational models for personality on Twitter corpus of 335 users was the (Quercia et al., 2011). They showed that a user's personality traits can be predicted only using three features : following, followers and listed counts.

Personality	Description
Openness	Imaginative, insightful and have wide interest
Conscientiousness	Organised, thorough and planned
Extroversion	Talkative, energetic and assertive
Agreeableness	Sympathetic, kind and affectionate
Neuroticism	Tense, moody and anxious

Table 1: Description for OCEAN Model

Schwartz Values - Societal Sentiment: The societal sentiment model introduced by Schwartz and Bilsky (1990) and modified by Schwartz (1992). The model defines ten basic and distinct personal ethical values (henceforth only values), that respectively are given in the table 2:

Values	Description
Achievement	sets goals and aims at achieving them
Benevolence	seeks to help others and provide general welfare
Conformity	obeys clear rules, laws and stuctures
Hedonism	seeks pleasure and enjoyment
Power	controls and dominates others, control resources
Security	seeks health and safety
Self-direction	wants to be free and independant
Stimulation	seeks excitement and thrills
Tradition	does things blindly because they are customary
Universalism	seeks peace, social justice and tolerance for all

Table 2: Description for Schwartz Values

The computational Schwartz model has been first proposed by (Maheshwari et al., 2017). The authors released a corpus of 367 unique users having 1,608 average tweets per user labelled with values traits. The highest number of tweets for one user was 15K, while the lowest number of tweets for a user was a mere 100.

3.1 Psycholinguistic and Network Features

Several different types of features are used depending upon classifiers. An exhaustive set of features include – (**f1**) Word N-grams; (**f2**) POS tags; (**f3**) Lingustic Features (LIWC[1]; Harvard General Inquirer, MRC psycholinguistic feature; Sensicon[2]); (**f4**) Network properties (network size, betweenness centrality, density and transitivity); (**f5**) speech-act classes; (**f6**) sentiment amplifiers (exclamation marks, quotes, ellipses, interjections, emoticons, word/sentence length); (**f7**) misspelt words (SMS slang, stressed words, capitalized words, wrong spellings); (**f8**) presence of *umm/bint* or *abu* in username (a common suffix for women and men respectively in Arabic); (**f9**) Sentiment/Emotion lexica (NRC emotion Lexicon (Mohammad et al., 2013), Sentiwordnet (Baccianella et al., 2010)); (**f10**) Topics words obtained from topic model. A brief overview about the Sociological models and features used are illustrated below.

3.2 Building Classifiers and Performance

We collected data from several sources to build five classification models. Here, for each model, we report the best classifier. All the results reported in Table 3 are based on 10-fold cross validation on the respective dataset. **Personality**: A SVM-based model outperforms the state-of-the-art (Verhoeven et al., 2013) by 10%, achieving average F-Score of 79.35%. **Values and Ethics**: A SVM-based values classifiers achieves an average F-Score of 81%. Features used in this model (both personality and values) are reported in Table 3.

Features	Model	F-Score (SVM)	F-Score (LR)	F-Score(RF)
Lexicon	Personality	0.78	0.62	0.65
	Values	0.74	0.59	0.62
+Non-Linguistic	Personality	0.79	0.66	0.68
	Values	0.76	0.61	0.65
+Speech-Act	Personality	**0.80**	0.70	0.71
	Value	**0.81**	0.63	0.67

Table 3: Performance of Personality and Values Models.

4 Semantic Interpretation of Communities

A community in a social network is considered to be a group of nodes densely connected internally and sparsely connected externally. In this paper, we attempt to understand whether individuals in a community possess similar personalities, values and ethical background.

In order to analyze the behaviour of optimists/pessimists at societal level, the egocentric twitter network released by SNAP is used. The Twitter network, released by SNAP (Leskovec and Krevl, 2015) (nodes: 81,306, edges: 1,768,149) has been used to study community structure. We considered 1,562 ground-truth communities (after discarding communities having size less than 5 and with tweets less than 100).

In order to analyze whether people within the same community tend to be homogeneous with respect to their background values/ethics, we measure Shannon's Entropy (measure of the uncertainty) (Lin, 1991) for each dimension separately.

Higher entropy scores suggest lower similarity. To calculate the entropy score vector $X_{(i)}$ for a community $C_{(i)}$ consisting of n users as $u_{(1)}, u_{(2)}, u_{(3)}...u_{(n)}$, a matrix $A_{(i)}$ is created where $A_{(i,j)}$ row vector represents the estimated scores of each of the ten values for a user $u_{(j)}$ and $A_{(i,:,k)}$ column vector represents the estimated scores of k^{th} class for all n users. The $A_{(i,:,k)}$ column vector was transformed to a probability distribution vector $N_{(i,:,k)}$ using softmax-normalization:

[1] http://www.liwc.net/

[2] https://hlt-nlp.fbk.eu/technologies/sensicon

$$N_{(i,j,k)} = \frac{exp(A_{(i,j,k)})}{||exp(A_{(i,:,k)}||_1} \quad (1)$$

The entropy score $X_{(i,k)}$ for $N_{(i,:,k)}$ can be calculated using the following formulation:

$$X_{(i,k)} = -\sum_{j=1}^{n} N_{(i,j,k)} * \log N_{(i,j,k)} \quad (2)$$

	AC	BE	CO	HE	PO	SE	SD	ST	TR	UN
$u_{(1)}$	0.91	0.47	0.02	0.07	0.32	0.24	0.65	0.78	0.94	0.10
$u_{(2)}$	0.97	0.40	0.49	0.50	0.56	0.83	0.62	0.73	0.04	0.08
$u_{(3)}$	0.99	0.75	0.50	0.72	0.38	0.60	0.75	0.02	0.57	0.62
$u_{(4)}$	0.77	0.44	0.40	0.16	0.19	0.55	0.73	0.08	0.53	0.25
$u_{(5)}$	0.29	0.02	0.26	0.56	0.41	0.23	0.95	0.02	0.79	0.86
$X_{(i)}$	1.54	1.40	1.40	1.39	1.55	1.50	1.59	0.99	1.42	1.28
$S_{(i)}$	0.87	-0.12	-0.12	-0.19	0.95	0.57	1.26	-2.35	0.00	-0.87
$T_{(i)}$	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0

Table 4: Illustrates entropy calculation for values model. Here $T_{(i)}$ represents the binary estimate of fuzzy distribution of values and $S_{(i)}$ represents the zero-mean unit-variance scaled values of $X_{(i)}$ for a community $C_{(i)}$. Similarly, binary estimates for five personality traits $P_{(i)}$ of user $u_{(i)}$ are calculated.

After normalization, $N_{(i,:,k)}$ vector represents the probability distribution of k^{th} value class across n users where entropy score $X_{(i,k)}$ represents the randomness in community along k^{th} value class. The lower the randomness, higher the k^{th} class is dominant in the $C_{(i)}$ community. Now, in order to obtain binary estimates $T_{(i)}$ for each of the ten values and classes in $C_{(i)}$ community, the entropy score vector $X_{(i)}$ is scaled using zero-mean unit-variance method and for numerical values greater than 0, 1 was assigned and for numerical values less than 0, 0 was assigned as class label for $C_{(i)}$ community. Instead of labelling a community $C_{(i)}$ with a class having minimum entropy, the scaling approach is used for the purpose of preserving the fuzzy distribution of values at community level. The obtained $T_{(i)}$ vector represents the fuzzy distribution of values and is thus a representation to capture the semantic information about the community. Having built the model and the classifiers we now try to understand the Psycho-Sociological aspects of the mention network in Twitter.

5 Understanding Psycho-Sociological Aspects of Twitter Mention Network

Network Creation: From the obtained tweets of the SNAP dataset, community level communication networks were created by looking at the @ mention in users tweets. For the network creation, Gephi API[3] is used. In the networks each node represents a user in the network and the edge represents mention link. The users who are never mentioned by somebody and never mentioned someone else were discarded at this stage as they will not contribute anything in understanding the dynamics of intra-community mention network. Once networks were formed, we analyzed their detailed characteristics. For example, let us take into consideration the following network in Figure 1 in order to analyze the following parameters. The nodes in the network represent the users are in the network and the edges represents the connection between the users. It is also important to note that not all the users in the network might not be connected. For example, when we take Figure 1 into consideration we are able to find that there are a total of 10 users labeled from A to J and 6 nodes(users) are connected within the network whereas 4 nodes (users) are disconnected from the network. After we created the network we tried to understand the level of connectivity, after which we analyzed the community type and the personality type along with the mention pattern.

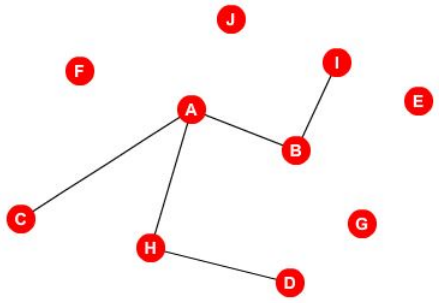


Figure 1: The network of users

Understanding the Level of Connectivity: To understand the dynamics of intra-community mention network we calculate user specific eigenvector centrality (Newman, 2008). Consider the centrality of vertex *i* could be denoted by x_i, could be calculated by making x_i proportional to the average of the centralities of i's network neighbors which is given in the below formula:

$$1/\lambda \sum_{j=1}^{n} A_{ij} x_j \quad (3)$$

, where λ is a constant. Let us assume the exam-

[3] https://gephi.org/

ple of Figure 1 here. In this case the eigen-vector centrality for each node [A, B, C,....I, J] is calculated. The centrality measure is one of the most fundamental measure in the network structure. It is used to determine the central node, and helps in identifying the centralized person who other people are connected with in the social network (Newman, 2008).

Community Type and Mention Patterns: After calculating the eigen-vector centrality at the user level i.e for each nodes, we calculated the eigen-vector centrality for each community. This analysis was done in order to determine how the users behave with each others within their own community. Now, lets say user u_i is connected with *n* number of users $[u_j, u_k, ..., u_n,]$ via mention links within a community. From Figure 1,we are able to notice that A is connected to C,H,B and I. D is connected indirectly to A via node H and hence we consider D only while calculating the eigen-vector centrality for node H. Now for user A, we calculate pair-wise eigen-vector centrality between each pair of users [(A, B), (A, C), (A, H), (A, I)] and obtain a vector for the user A. To get the final average intra-community connectivity score for user A, all the *n* scores are further averaged by dividing their sum by *n*. The score that is obtained now is the average connectivity of the user A within the community. Following this method we obtain connectivity score of all the users within the community and those scores are then further averaged by the total number of users (excluding users who never got mentioned or never mention someone else) within the community. This score obtained is the average score of the community for that particular user. Similarly, average connectivity scores are calculated for each community in the SNAP network and further those obtained scores are averaged based on community type (i.e., power, hedonic, etc.). These category-wise average connectivity scores are finally reported to understand the intra-community psycho-sociological aspects of the Twitter mention network.

Personality Type and Mention Patterns: To understand who is mentioning whom we consider user specific average eigen-vector centrality. For example, in the first iteration we take all the open type people and find out their average eigen-vector centrality over all communities. Then we further divide them into 5 classes (i.e., open, conscientious, etc.). Thus class-to-class average connectivity (i.e., open-open, open-conscientious, open-extrovert, open-agreeable, and open-neurotic) was obtained. This process was repeated for all other personality types.

Finally, for more granular understanding the eigen-vector centrality scores were divided into three bands - high, mid, and low. These values were then scaled between 0-100 by looking at the overall connectivity score (high, low) distribution at corpus level as shown in Figure 3.

6 Obtained Mention Network Patterns

We present our findings in three parts. First part details psycho-sociological patterns of the mention network, whereas the second part tries to answer the question – *who-mentions-whom*? Finally we try to analyze the relationship between closeness and reciprocity of the community with different community sizes.

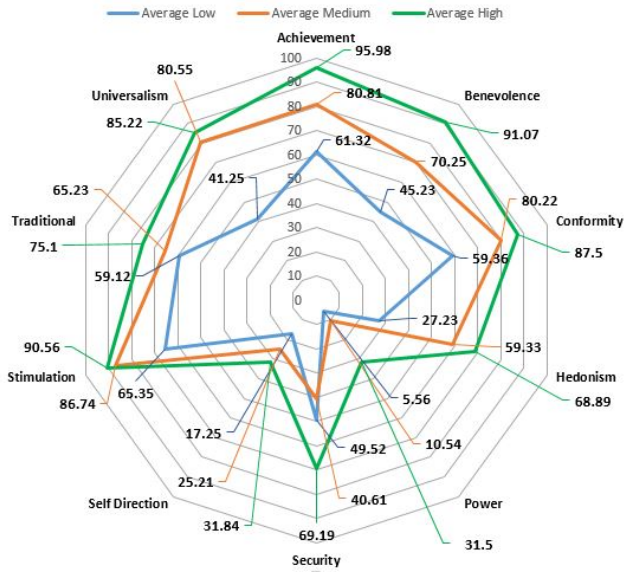
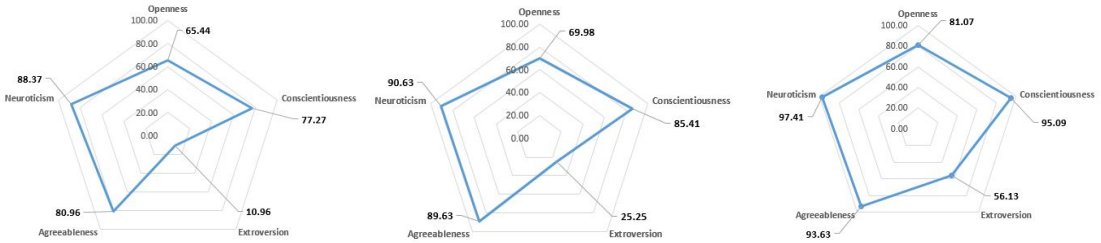


Figure 2: Communication within each community of the values model

Psycho-Sociological Patterns of Mention Network: Figure 2 reports obtained results of our empirical analysis. Results indicate that members from stimulation, achievement, and benevolent communities are closely connected among themselves while the members in other communities do not show significant connectivity among themselves.

We also observe that people who are independent, i.e, self-directed do not involve much in connecting themselves with other members in their



(a) Low connectivity (b) Medium connectivity (c) High connectivity

Figure 3: Psycho-sociological patterns of Twitter mentions network

community. Security oriented people i.e., those who follow strict rules and regulations are found to have reasonably balanced connectivity. People belonging to the traditional groups who follow rituals blindly, are loosely connected to the world. One significant observation was that the universal people, who are the people tending to be more inclined towards social justice and tolerance show high connectivity with other people in the community. Further analysis reveal that in general universal people tend to unicast (i.e., @mention someone specific) messages rather than broadcast , which is justifiable to their nature. The members of power-oriented communities are those who seek to dominate other people in their community and hence the communication is low among these people.

Node	Closeness
A	0.40
B	0.31
C	0.25
D	0.21
E	0.0
F	0.0
G	0.0
H	0.31
I	0.21
J	0.0

Table 5: Closeness Centrality for Figure 1

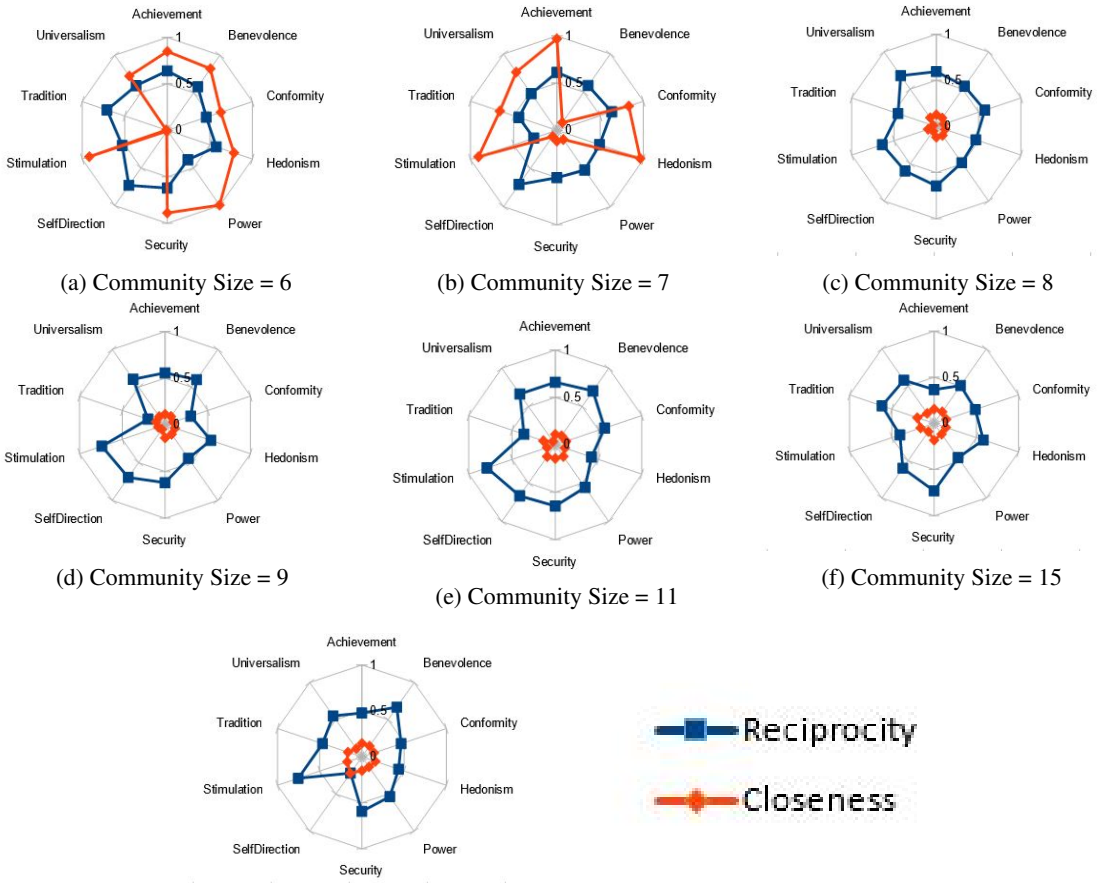
Who-Mentions-Whom: Results of this analysis is reported in Figures 3a, 3b and 3c. The result is presented in three sets i.e., low, medium, high connectivity. We notice that in the highly connected communities, the neurotic people who are mostly tense, moody, anxious are more connected to other people in the network. Agreeable and conscientious people also maintain a good relationship with others. In the case of medium and low connected communities , the neurotic people tend to maintain good connectivity with others than people possessing other values. We also infer that extroverts (high,mid and low) also tend to broadcast their messages rather than sending it to someone specific, therefore their connectivity is low in the mention network.

Similar practice has also been noticed by open people in low connected group. Therefore, we can conclude universal, extrovert, and open people prefer broadcasting over unicasting messages (via @ mention)

Closeness vs Reciprocity: In a connected graph,the closeness or the closeness centrality of a node is used to measure how close a particular node is with respect to other nodes in the network. It is calculated as the sum of the lengths of the shortest paths between the particular node and the other nodes in the graph. It can be seen that, the more central the node is, the more closer it is to the other nodes.

For example, in Figure 1 we are able to find that node A is more closely associated with other nodes and has the highest closeness centrality among the other nodes. It is also observed from Table 5 that those nodes which are not connected in the network shown in Figure 1 are having 0 as their closeness centrality measure.

The social norm of reciprocity is the expectation that people will respond to each other in similar ways. Therefore in a mention network, if a particular user mentions another user in his/her tweet and the other user on the other hand mentions him/her back then we can find that reciprocity can be achieved between those two users. The result is provided as an analysis of closeness vs reciprocity for various sizes of the community in Figure 4. From the analysis, we observe that as the size of the community increases the reciprocity de-



(a) Community Size = 6

(b) Community Size = 7

(c) Community Size = 8

(d) Community Size = 9

(e) Community Size = 11

(f) Community Size = 15

(g) Community Size = 16

Figure 4: Closeness vs. Reciprocity for various community sizes

creases and the closeness increases. This is because the closeness centrality is calculated for an entire network and hence as the size of the network increases the closeness tend to increase.

Reciprocity on the other hand is calculated for the mentions network and as the size of the network increases there is a high chance that two users do not mention each other in their tweets. Let us consider one example from our analysis, here we consider the self-directed community of people of community size 15 and 16 respectively. From Figure 4 we are able to find that as the size of the community increases from 15 to 16 the reciprocity decreases and the closeness increases.

7 Conclusion and Future Work

This paper presents an empirical analysis to understand the psycho-sociological aspects of Twitter mentions network within a social network community. Here, we take the explanatory approach; however, we strongly believe that the obtained empirical results could be further used to predict individual/group communication behavior. We would be woking on finding similar patterns in (i) Twitter favorite network i.e., *who has liked whom*, and (ii) Retweets network i.e., *who retweets whose tweet*. We would also like to understand inter-community mention network pattern – i.e., is there any selectional preference when someone chooses to communicate with someone outside his/her community? We believe that this kind of models may become extremely useful in the future for various purposes like Internet advertising (specifically social media advertising), community detection, computational psychology, recommendation systems, sociological analysis over social media.Now that we have calculated various measures and obtained analytical results. In the future, based on these results given a community we try to predict who will mention whom?

References

Andrea Apolloni, Karthik Channakeshava, Lisa Durbeck, Maleq Khan, Chris Kuhlman, Bryan

Lewis, and Samarth Swarup. 2009. A study of information diffusion over a realistic social network model. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 675–682. IEEE.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

Fabio Celli, Fabio Pianesi, David Stillwell, and Michal Kosinski. 2013. Workshop on computational personality recognition (shared task). In *Proceedings of the Workshop on Computational Personality Recognition*.

Fabio Celli, Bruno Lepri, Joan-Isaac Biel, Daniel Gatica-Perez, Giuseppe Riccardi, and Fabio Pianesi. 2014. The workshop on computational personality recognition 2014. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1245–1246. ACM.

David Roxbee Cox and David Oakes. 1984. *Analysis of survival data*, volume 21. CRC Press.

Lewis R Goldberg. 1990. An alternative" description of personality": the big-five factor structure. *Journal of personality and social psychology*, 59(6):1216.

Manuel Gomez Rodriguez, Jure Leskovec, and Bernhard Schölkopf. 2013. Structure and dynamics of information pathways in online media. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 23–32. ACM.

Masahiro Kimura, Kazumi Saito, Ryohei Nakano, and Hiroshi Motoda. 2010. Extracting influential nodes on a social network for information diffusion. *Data Mining and Knowledge Discovery*, 20(1):70–97.

Jure Leskovec and Andrej Krevl. 2015. {SNAP Datasets}:{Stanford} large network dataset collection.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151.

Tushar Maheshwari, Aishwarya N Reganti, Upendra Kumar, Tanmoy Chakraborty, and Amitava Das. 2017. Semantic interpretation of social network communities. In *AAAI*, pages 4967–4968.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*.

Mark EJ Newman. 2008. The mathematics of networks. *The new palgrave encyclopedia of economics*, 2(2008):1–12.

Daniele Quercia, Michal Kosinski, David Stillwell, and Jon Crowcroft. 2011. Our twitter profiles, our selves: Predicting personality with twitter. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 180–185. IEEE.

Daniel M Romero, Brendan Meeder, and Jon Kleinberg. 2011. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 695–704. ACM.

Shalom H Schwartz and Wolfgang Bilsky. 1990. Toward a theory of the universal content and structure of values: Extensions and cross-cultural replications. *Journal of personality and social psychology*, 58(5):878.

Shalom H Schwartz. 1992. Universals in the content and structure of values: Theoretical advances and empirical tests in 20 countries. *Advances in experimental social psychology*, 25:1–65.

Shalom H Schwartz. 2012. An overview of the schwartz theory of basic values. *Online readings in Psychology and Culture*, 2(1):11.

Ben Verhoeven, Walter Daelemans, and Tom De Smedt. 2013. Ensemble methods for personality recognition. In *Proceedings of the Workshop on Computational Personality Recognition*, pages 35–38.

Mudasir Wani and Manzoor Ahmad. 2014. Survey of information diffusion over interaction networks of twitter. *International Journal of Computer Application*, 3(4):310–313.

Jiang Yang and Scott Counts. 2010. Predicting the speed, scale, and range of information diffusion in twitter. *ICWSM*, 10:355–358.

Tauhid R Zaman, Ralf Herbrich, Jurgen Van Gael, and David Stern. 2010. Predicting information spreading in twitter. In *Workshop on computational social science and the wisdom of crowds, nips*, volume 104, pages 17599–601. Citeseer.

Study on Visual Word Recognition in Bangla across Different Reader Groups

Manjira Sinha
Conduent Labs India
Bangalore, India

Tirthankar Dasgupta
TCS Innovation Labs
Kolkata, India

Anupam Basu
IIT Kharagpur
Kharagpur, India

{manjira87, iamtirthankar, anupambas}@gmail.com

Abstract

This paper presents a psycholinguistic study of visual word recognition in Bangla. The study examines the relationship among different word attributes and word reading behaviors of the two target user groups, whose native language is Bangla. The different target user groups also offer insights into the subjectivity of written word comprehension based on the readers background. For the purpose of the study, reading in terms of visual stimulus for word comprehension has been considered. To the best of the knowledge of the authors, this study is the first of its kind for a language like Bangla.

1 Introduction

Recognition and understanding of words are basic building blocks and the first step in language comprehension. At this stage, the form (visual representation) joins the meaning (conceptual representation). Therefore, the cognitive load associated with word reading is a significant contributor to the overall text readability. The present study aims to capture the salience effects of different word attributes on the word reading performance in Bangla, the second most spoken (after Hindi) and one of the official languages of India with about 85 million native users in India [1] . The features studied in this work, encompass orthographic properties of a word like length in terms of the number of visual units or akshars; number of unique orthographic shapes i.e, the characteristic strokes and complexity measures based on the familiarity of the akshars and strokes in a word. Phonological properties of a word such as number of syllables and spelling to sound consistency have also been taken into account along with the semantic attributes of a word like number of synonyms and number of senses. Moreover, the feature list also includes word collocation attributes such as orthographic neighborhood size and phonological neighborhood size, which situate the given word with respect to other members in the vocabulary. The effects of the word attributes have been measured in terms of the reaction time and performance accuracy data obtained from empirical user experiments.

The paper is organized as follows: Section 2 presents the relevant literature study, section 3 describes the participants details; section 4 and 5 states data preparation and the psycholinguistic experiment respectively; section 6 presents the feature descriptions and the experimental observations against words and non-words; finally, section 7 concludes the paper.

2 Related Works

Research in word recognition has been central to many areas in cognitive-neuroscience (Frost et al., 2005), educational processes (Seidenberg, 2013), attention (Zevin and Balota, 2000), serial versus parallel processing (Coltheart et al., 1993), connectionism (Plaut et al., 1996) and much more. Typically, two different techniques are used to study visual word recognition: the lexical decision tasks and the naming task (Balota et al., 2004). In lexical decision task, a letter string is presented to a participants are asked to decide whether the given string is a valid word in their language. On the other hand, in the naming task, participants are asked to read allowed a letter string as quickly as possible. The time taken by a subject to complete each task after the visual presentation of the target is defined as the response time (RT). An analysis of the reaction times of the subjects

[1] http://www.ethnologue.com/statistics/size

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 427–434

reveals the actual processing of words in brain. The early works in word recognition involves two distinct models: the activation model or the logogen model (Morton, 1969) and the search model (Forster and Bednall, 1976); both of these two models are based on the fundamental premises of the frequency effects in word recognition. The frequency effect in word recognition claims that the high frequency words are recognized more accurately and quickly than the low-frequency words (Murray and Forster, 2004). The logogen model assumes recognition of words in terms of the activation of the constituent linguistic features (called the logogens). Each logogen has got a base activation value (also called the resting activation) that facilitates the recognition process. The resting activation of a given logogen is determined by its frequency of occurrence. That is, high frequency words have higher base activation value than the low frequency words. The search model, on the other hand, assumes that words are organized according to their frequencies and are searched serially. (Taft and Hambly, 1986) have a proposed hybrid model that includes features of both the activation and serial search process. The interactive activation (IA) model (Diependaele et al., 2010) follows the connectionist approach and also incorporates the logogen model. In this framework, a word is initially perceived via the basic orthographic, features which in turn activate the higher level syntactic and semantic features. The IA model also accounts for the word superiority effect that assumes alphabets are recognized more accurately and quickly when they occur in a word as compared to a non-word (Grainger and Jacobs, 1996). An important extension of the IA model is the dual-rout cascaded (DRC) model (Coltheart et al., 2001). This model assumes two parallel process of word recognition: the lexical route and the sub-lexical route. The lexical route accounts for the recognition process through the parallel activation of the orthographic and phonological features of a word. On the other hand, the sub-lexical route possesses a serial processor that converts graphemic representations into phonemic forms. As an alternative to two different processing paths n the DRC model, the parallel distributed processing model (PDP) (Seidenberg and McClelland, 1989) has proposed a single architecture to explain different processing outputs. The model incorporate the distributed nature by assuming that each word is associated with some distinct activation pattern across a common set of features used to recognize the word. The features may include, orthography, phonology, morphology or semantic. Generalizations of the PDP model for non-words and irregular words have been proposed by (Plaut et al., 1996)

3 Participants

In order to understand how the different cognitive processes vary across different user groups, two categories of users have been considered for each user study. Group 1 consists of 25 native users of Bangla in the age range 21-25 years, who are pursuing college level education and group 2 consists of 25 native users in the age range 13 to 17 years (refer to figure 1). In this paper, the variations in age and years of education have been taken into account. Moreover, we have considered a distribution over medium to low socio-economic sections with monthly household income ranges INR 4500 to INR 15000. The Socio-Economic Classification (SEC) has been performed according to the guidelines by the Market Research Society of India (MRSI) [2]. MRSI has defined 12 socio-economic strata: A1 to E3 in the decreasing order. The containment of the socio-economic range was necessary as it directly affects education, literacy and thus the state of comprehension skills of a reader. In addition, to capture the first-language skill, each native speaker was asked to rate his/her proficiency in Bangla on a 1-5 scale (1: very poor and 5: very strong), see figure 2.

Type	Background	Mean age (Standard deviation)
Group 1 (adult): 25 native speakers of Bangla	Education: pursuing graduation	22.8 (1.74)
	Socio Economic Classification: B2-D2	
Group 2 (minors): 25 native speakers of Bangla	Education: pursuing school education	15 (1.24)
	Socio Economic Classification: B2-D2	

Figure 1: Participants' details

[2]http://imrbint.com/research/The-New-SEC-system-3rdMay2011.pdf

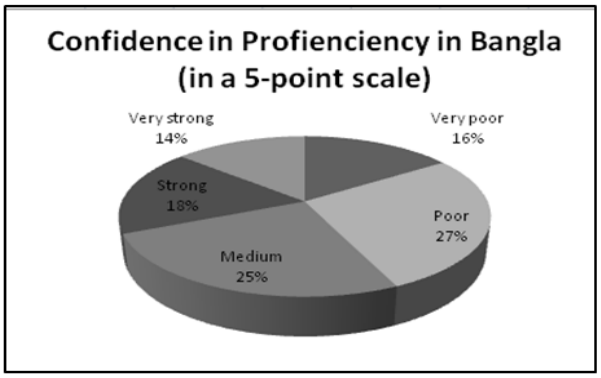


Figure 2: Proficiency in the mother tongue

4 Data preparation

From a Bangla corpus [3] of about 400,000 unique words, we have sampled 3500 words for the study. The words were selected in such a way that they represent the 'average' words over the corpus. The median values of word frequency distribution and length distribution lie at 368 and 5 respectively (refer to figur 3 for some sample words used in experiment). In a psycholinguistics, to preserve the experimental standard, it is essential to restrict the participants from making any strategic guess about the input stimuli. This has been achieved by randomly introducing non-words in between the valid words during the experiment. However, designing non-words are a non-trivial process, and often the reader's response to the different types of non-words opens up new insights into the process of word comprehension. Some examples of non-words are provided in figure 4.

5 Experimental Procedure

We have conducted lexical decision task (LDT) experiment (Meyer and Schvaneveldt, 1971) to study the visual recognition of Bangla words by native speakers. In this experiment, a participant is presented with a visual input, generally, a string of letters that can be words, non-words or pseudo words. Their task is to indicate, whether the presented stimulus is a valid Bangla word or not. The reaction time against each participant and the accuracy against each experimental stimulus across all the participants are recorded for further analysis. The time window for a user to submit any response has been set at 4 seconds, failing that a No Response is recorded. In either cases it is followed by hash signs (####) followed by the next letter string with 2.5 second delay. No response against a stimulus is automatically recorded as wrong response by the user.

Fifty users from the two target user groups participated in the LDT experiment. The 5000 experimental words (2500 words and 2500 non-words) were distributed randomly among 67 equal sized 75-word sets. Each user was presented maximum of three sets a day with at least one hour gap between two sets. Before recording experimental data, a sample set made up of 20 words was presented to the users to make them accustomed with the experiment.

6 Observations

All the incorrect responses and extreme reaction times (RT: the time taken to respond to a stimuli) have been discarded. Participants and experimental words having less than 70% accuracy have also been discarded. Finally, 440 words with RT of 42 (22 from group 1 and 20 from group 2) participants have been used for further study.

The RTs of each user have been normalized by z-transformation (Balota et al., 2007). The mean z-score over all users for a word has been computed. Negative z-scores indicate shorter response latencies. Paired t-test has been performed between results of the two user groups and $p < 0.05$ has been found signifying the difference between reading characteristics of the two user groups. Next, we have studied the influence of different word features on the outcome of the lexical decision task. The word features studied in this paper have been selected based on their prominence in the literature (Yarkoni et al., 2008) and their relevance with respect to Bangla. The features are:

•**Morphological Family size:** The morphological family size of a word **w** comprises of all the inflected, derived and compound paradigms that contains the word *w* (De Jong IV et al., 2000).

•**Word length (linear):** The length is measured in terms of the number of visual units or akshars; as Bangla belongs to the abugida group, mere alphabetic word length does not reflect the difficulty encountered in reading (Sinha et al., 2012b).

•**Number of complex characters in a word:** Complex characters are the consonant conjuncts or *jukta-akshars* present in a word.

•**Number of unique shapes in a word:**

[3]The Unicode corpus of Bangla was developed by the authors as a part of a broader study, the details are not in scope of this paper.

Category	Word (frequency, length in akshars, number of unique features)
High frequency long words (HL)	পরিবার (*paribAra*, family) (13403, 4, 6); আন্তর্জাতিক (*AntarjAtika*, international)(6936, 5, 13); তথ্যপ্রযুক্তি(*tathyaprayukti*, information technology) (2332, 5, 13), উল্লেখযোগ্য (*ullekhayogya*, worth mentioning)(1143, 5, 13)
High frequency short words (HS):	লঙ্ঘন (*la.nghana*, to cross) (541, 3, 7); অঞ্চল (*a~ncala*, region)(2163, 3, 8); বাড়ি (*bA.di*, house) (37984, 2, 6); কেন্দ্রীয় (*kendriya*, central) (22465, 3, 11) ;
Low frequency long words (LL):	নির্বাচকমণ্ডলী (*nirbAchakamandali*, selection committee) (74, 7, 13); স্বাতন্ত্র্যবোধ (*sbAtantrabodha*, feeling of freedom)(3, 5, 10);
Low frequency short words (LS):	দপ্তর (*daptara*, office) (10, 3, 8); পীড়িত (*pI.dita*, sufferer) (95, 3, 9); শিকার (*shikara*, hunt) (73,3, 7)

Figure 3: Examples of valid-words for experiment [4]

Type	Example
Proper non-word	ণজতথী (*NajatathI)*
Non-words from valid words	
Character deletion	রাজকীয় (*rAjakiYa*, royal) > রাজকী
Character insertion	রন্ধনশালা(*randhanshAlA*, kitchen) > রন্ধনশালাম
Jumbled	সংবাদপত্র (*sa.NbAdapatra*, newspaper) > সংদপবাত্র
Substitution by similar orthographic or phonological unit	তারিফ (*tAripha*, praise)> তারিপ (*tAripa*), চালান (*chAlAna*, transaction)> টালাণ (*TAlANa*)

Figure 4: Construction of non-words for experiment

Bangla script uses the space in a non-linear way and the akshars hangs from a distinct horizontal head-stroke called mAtrA. The letters are made up of combinations of different shapes or strokes. All together 57 unique strokes have been identified and indexed accordingly. The initial hypothesis is that more the number of distinct shapes in a word; the more difficult it is to comprehend.

•**Orthographic word complexity:** During visual word recognition, the reader has to recognize the orthographic patterns (Selfridge, 1958). Word level representations interact with the letter level representations i.e, the characteristic shapes or strokes (refer to). As no standard dataset on shape combinations in Bangla letters is available, the unique shapes or strokes have been identified intuitively across all the Bangla letters including the consonant conjuncts. The Bangla Akademi font has been considered as standard Bangla orthography. All together 57 unique strokes have been identified and numbered. Every Bangla letter has been represented as a combination of the constituent shapes. To capture the interactive nature

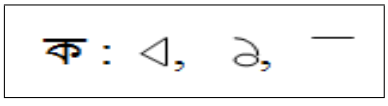


Figure 5: characteristics strokes of Bangla akshars

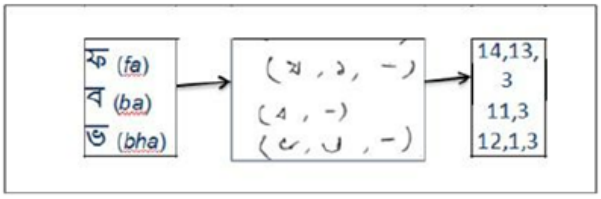


Figure 6: Mapping of Bangla akshars to characteristics shapes

of visual complexity, an orthographic complexity

model has been derived in the following way:

(a) The difficulty (d(s)) of a characteristic shape (i) or stroke is inversely proportional to its familiarity or frequency (f(s)). The frequency of the shapes has been calculated from the unique word list of the Bangla corpus without considering the frequency of each word.

$$d(s) = 1/f(s). \quad (1)$$

(b) The difficulty (d(a)) of an akshar (a) depends on the sum of the complexity of its shapes normalized by the number of shapes (n)

$$d(a) = 1/n \sum_i d(s_i) \quad (2)$$

(c) Finally, the difficulty (d(w)) of a word (w) is the sum of the complexity of its constituent akshars normalized by word length (l) and multiplied by the inverse of the word frequency (f(w))

$$d(w) = 1/f(w)\Big[1/l\big(\sum_j d(a_j)\big)\Big] \quad (3)$$

•**Orthographic & phonological neighborhood:** We have constructed akshar based, orthographic shape based and phonological pattern based neighborhood structure. The akshar based distance measure treats all akshars as of same visual complexity regardless of their orthographic properties, this is the reason distance among words based on orthographic strokes has been treated separately. At each level of orthographic information, the neighbors have been categorized into three groups based on their distance from the given word.

•**Number of syllables:** The syllabification of the Bangla words has been performed using a Bangla Grapheme to Phoneme conversion tool, developed inhouse.

•**Semantic neighborhood:** This measure represents the number of semantic neighbors of a word within the lexical organization of the language. This is computed from the semantic lexicon described in (Sinha et al., 2012a).

The mean and standard deviation values of the word features described above have been presented in figure 7. We have analyzed the RT corresponding to the above features using Spearman's correlation coefficient. The coefficient values between each word attribute and word recognition performance for the two user groups have been presented in figure 8.

From 8 we can observe that the correlation coefficients values for lexical decision latencies and decision accuracies are always less than 0.5, though they are different for different groups. The difference in the coefficient values may be attributed to the different reading patterns of the two groups. Number of syllables has similar correlation coefficients as word length because most often the akshars boundaries match the phonological syllable boundaries. The measure of orthographic word complexity possess low correlation coefficients with reaction times and accuracies, this can be an outcome of considering only the orthographic attributes of a word, isolating it from the phonological or semantic dimensions. In future, the measure needs to be augmented with those word features.

Number of unique shapes and complex characters also do not show significant correlation. Spelling to sound consistency also has a moderate correlation with the groups. This shows that speakers are not much sensitive towards the minor inconsistencies in spelling to sound mapping. The correlation coefficients of distant orthographic and phonological neighbors, immediate orthographic neighbors at shape level and semantic neighborhood are not significant for both groups. These indicate that after a threshold distance, the similarity or dissimilar-ity of the given word with other words in vocabulary does not affect the readers decisions. In addition, at shape level, the number of immediate orthographic neighbors may be unimportant due to the fact that often an akshar is constituted with more than 2 characteristic orthographic shapes and therefore, while reading, such minor changes in orthographic properties may go unnoticed.

Finally, the present calculation of semantic neighborhoods has been based on exhaustive language information (Sinha et al., 2012c), but the actual users may not possess such deep language knowledge and therefore are less affected by the semantic neighborhood structure. On the other hand, the number of senses or meaning of a word does not have inhibitory effect on the decision making process as the no ambiguity had to be resolved here, instead the use of a word in different

No.	Feature	Mean	Standard Deviation
1	Word frequency (log base 10)	1.81	1.09
2	Morphological family size	2.36	1.27
3	Word length (in akshar)	4.48	1.29
4	Number of complex characters	0.71	1.35
6	Number of unique shapes	10	3
7	Orthographic word complexity	-4.04	0.29
8	Immediate orthographic neighbors (akshar level)	2.11	0.16
9	Close orthographic neighbors (akshar level)	3.65	0.27
10	Distant orthographic neighbors (akshar level)	4.23	0.34
11	Immediate orthographic neighbors (shape level)	1.96	0.17
12	Close orthographic neighbors (shape level)	3.47	0.25
13	Distant orthographic neighbors (shape level)	4.71	0.41
14	Number of syllables	3.60	1.10
15	Immediate phonological neighbors	2.32	0.17
16	Close phonological neighbors	3.71	0.31
17	Distant phonological neighbors	4.33	0.36
18	Spelling to sound consistency	0.67	0.21
19	Number of senses of a word	0.13	0.06
20	Semantic neighborhood	18	6

Figure 7: Properties of valid words for experiment

contexts have increased its chance of encountering with the native readers of Bangla more often.

Moreover, the decisions against non-words are equally interesting to the decisions against the valid words. Non-words such as *kakShataNa* [correct: (katakShaNa, time duration)], *AkampIta* [correct: (akampita, steady)] and *TAlAN* [correct: (cAlAna, transaction)] have almost always been perceived as correct words by the readers due to their orthographic and phonological proximity to the correct words. On the other hand, proper non-word i.e, an arbitrary letter string such as *NajatathI* has been accurately classified as invalid. This indicates that the cognitive processes of reading are sensitive to the probability of what akshar pattern can occur in a valid Bangla word.

7 Conclusion

In this paper, we have presented a study on the comprehension difficulty of visual word recognition in Bangla stored as a lexical decision database. Number of interesting observations has been made from the experimental data and the observations have been complemented with rational inferences based on them. The correlation coefficients among word attributes and reaction time data has revealed that individually no feature has a large covariance factor, but the collective effect of all of them determines the cognitive load for comprehension. Moreover, using a reference language corpus based only on text from printed sources has proven to be a short-coming for drawing meaningful inferences. Some initial insights on the decisions corresponding to the non-words have also been presented.

References

David A Balota, Michael J Cortese, Susan D Sergent-Marshall, Daniel H Spieler, and MelvinJ Yap. 2004. Visual word recognition of single-syllable words. *Journal of Experimental Psychology: General*, 133(2):283.

D.A. Balota, M.J. Yap, K.A. Hutchison, M.J. Cortese, B. Kessler, B. Loftis, J.H. Neely, D.L. Nelson, G.B. Simpson, and R. Treiman. 2007. The english lexicon project. *Behavior Research Methods*, 39(3):445–459.

M. Coltheart, B. Curtis, P. Atkins, and M. Haller. 1993. Models of reading aloud: Dual-route and parallel-distributed-processing approaches. *Psychological Review; Psychological Review*, 100(4):589.

Max Coltheart, Kathleen Rastle, Conrad Perry, Robyn Langdon, and Johannes Ziegler. 2001. Drc: a dual

Feature		Correlation coefficients (r)[significance p-value<0.05]			
No.	Name	Group 1		Group 2	
		RT-z	Accuracy	RT-z	Accuracy
1	Word frequency (log base 10)	-0.43	0.16	-0.39	0.19
2	Morphological family size	0.04	0.03	0.04	0.02
3	Word length (in akshar)	0.35	-0.09	0.39	-0.07
4	Number of complex characters	0.27	-0.16	0.35	-0.23
6	Number of unique shapes	0.28	-0.05	0.37	-0.18
7	Orthographic word complexity	0.19	-0.13	0.21	-0.17
8	Immediate orthographic neighbors (akshar level)	0.13	-0.14	0.17	-0.09
9	Close orthographic neighbors (akshar level)	0.04	0.02	-0.15	0.13
10	Distant orthographic neighbors (akshar level)	0.03#	0.02#	-0.04#	-0.13#
11	Immediate orthographic neighbors (shape level)	0.04#	0.01#	-0.11#	-0.07#
12	Close orthographic neighbors (shape level)	0.03	0.01	-0.14	-0.09
13	Distant orthographic neighbors (shape level)	0.03	0.02	-0.12	-0.06
14	Number of syllables	0.36	-0.11	0.37	-0.18
15	Immediate phonological neighbors	0.22	-0.11	0.19	-0.12
16	Close phonological neighbors	0.04	0.03	0.03	-0.07
17	Distant phonological neighbors	0.06#	0.02#	0.11#	0.08#
18	Spelling to sound consistency	-0.23	0.09	-0.34	0.13
19	Number of senses of a word	-.37	0.21	-0.41	0.23
20	Semantic neighborhood	-0.23#	0.09#	-0.32#	0.07#

Figure 8: Correlation analysis between word attributes and data from LDT (correlation coefficients marked with # are not significant (p-value $>$ 0.05)

route cascaded model of visual word recognition and reading aloud. *Psychological review*, 108(1):204.

Nivja H De Jong IV, Robert Schreuder, and R Harald Baayen. 2000. The morphological family size effect and morphology. *Language and cognitive processes*, 15(4-5):329–365.

K. Diependaele, J.C. Ziegler, and J. Grainger. 2010. Fast phonology and the bimodal interactive activation model. *European Journal of Cognitive Psychology*, 22(5):764–778.

Kenneth I Forster and Elizabeth S Bednall. 1976. Terminating and exhaustive search in lexical access. *Memory & Cognition*, 4(1):53–61.

Stephen J Frost, W Einar Mencl, Rebecca Sandak, Dina L Moore, Jay G Rueckl, Leonard Katz, Robert K Fulbright, and Kenneth R Pugh. 2005. A functional magnetic resonance imaging study of the tradeoff between semantics and phonology in reading aloud. *NeuroReport*, 16(6):621–624.

Jonathan Grainger and Arthur M Jacobs. 1996. Orthographic processing in visual word recognition: a multiple read-out model. *Psychological review*, 103(3):518.

David E Meyer and Roger W Schvaneveldt. 1971. Facilitation in recognizing pairs of words: evidence of a dependence between retrieval operations. *Journal of experimental psychology*, 90(2):227.

John Morton. 1969. Interaction of information in word recognition. *Psychological review*, 76(2):165.

Wayne S Murray and Kenneth I Forster. 2004. Serial mechanisms in lexical access: the rank hypothesis. *Psychological Review*, 111(3):721.

David C Plaut, James L McClelland, Mark S Seidenberg, and Karalyn Patterson. 1996. Understanding normal and impaired word reading: computational principles in quasi-regular domains. *Psychological review*, 103(1):56.

Mark S Seidenberg and James L McClelland. 1989. A distributed, developmental model of word recognition and naming. *Psychological review*, 96(4):523.

Mark S Seidenberg. 2013. The science of reading and its educational implications. *Language learning and development*, 9(4):331–360.

Oliver G Selfridge. 1958. Pandemonium: a paradigm for learning in mechanisation of thought processes.

M. Sinha, T. Dasgupta, and Basu A. 2012a. A complex network analysis of syllables in bangla through syllablenet. In Sobha L Girish Nath Jha, Kalika Bali, editor, *Workshop on Indian Language and Data: Resources and Evaluation, LREC*, pages 131–138, May.

M. Sinha, S. Sharma, T. Dasgupta, and Basu A. 2012b. New readability measures for bangla and hindi texts. Communicated in the 24th International Conference on Computational Linguistics,2012, IIT Bombay, August.

Manjira Sinha, Abhik Jana, Tirthankar Dasgupta, and Anupam Basu. 2012c. A new semantic lexicon and similarity measure in bangla. In *Proceedings of the 3rd Workshop on Cognitive Aspects of the Lexicon*, pages 171–182, Mumbai, India, December. The COLING 2012 Organizing Committee.

Marcus Taft and Gail Hambly. 1986. Exploring the cohort model of spoken word recognition. *Cognition*, 22(3):259–282.

T. Yarkoni, D. Balota, and M. Yap. 2008. Moving beyond coltheart?s n: A new measure of orthographic similarity. *Psychonomic Bulletin & Review*, 15(5):971–979.

Jason D Zevin and David A Balota. 2000. Priming and attentional control of lexical and sublexical pathways during naming. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 26(1):121.

Demystifying Topology of Autopilot Thoughts: A Computational Analysis of Linguistic Patterns of Psychological Aspects in Mental Health

Bibekananda Kundu and Sanjay Kumar Choudhury
Language Technology, ICT and Services
Centre for Development of Advanced Computing, Kolkata
E-mail: {bibekananda.kundu,sanjay.choudhury}@cdac.in

Abstract

The paper investigates topology of uncontrolled dynamic depressive thoughts which is popularly known as *"autopilot"* in the psychology domain. Persistent homology, a mathematical tool from algebraic topological has been applied on Vector Space representation of tweets generated by users having neurotic personality for determining the topological structure of autopilot thoughts. State-of-the-art machine learning techniques leveraging linguistic resources akin to LIWC, WordNet-Affect and SentiWordNet have been applied for identifying neurotic personality from different Twitter users. An initiative has been taken for empowering Neuro Linguistic Programming (Bandler and Grinder, 1975; Bandler and Grinder, 1979; Bandler and Andreas, 1985) and other psychotherapy techniques using Natural Language Processing in the domain of Mental Health.

1 Introduction

"Wherever there are sensations, ideas, emotions, there must be words"

— Swami Vivekananda.

We use language for thinking, experiencing, expressing, communicating and problem solving. So, to analyze one's thought process, language is a symbolic medium. In psychotherapy, language is considered as a primary tool to understand patients' experiences and express therapeutic interventions (Pennebaker et al., 2003). All psychological interventions rely on the power of language. Psychotherapists rarely intervene directly in their client's lives, they create changes in the thought process through conversation (Villatte et al., 2015). According to *Relational Frame Theory (RFT)* (Greenway et al., 2010), people use linguistic frames to understand the world around them, and subsequently solve problems. *RFT* has been suggested as an approach to understanding natural language systems. The theory lends itself well to assessment with Natural Language Processing (NLP) precisely because it relies on understanding interaction between sensation, affect, language, and behaviour. When someone uses language, they are labelling their experience. For example, someone might tweet *"I need to escape this world before I get crushed."* indicating a fear based affective response. We are planning to use NLP to assess this label (Pennebaker et al., 2015). Simply labelling events and their attributes as 'positive' or 'negative' increases associated memories and emotional salience. This type of relational network can be evoked with any number of internal or external stimuli, triggering the aforementioned internal feedback loop, and leading to psychological distress. For example, describing a 'negative' event such as a trauma can evoke intense fear and sadness and subsequent sobbing (Miner et al., 2016; Althoff et al., 2016). The person suffering from distress actually using a model of world which is very limited and in this world he/she find no appropriate choice from the options available to their model of world (Bandler and Grinder, 1975; Bandler and Grinder, 1979; Bandler and An-

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 435–446

dreas, 1985). Therefore, there is a requirement of expanding the model of world i.e. improvise the model to a better model which has more options. Therefore, the therapeutic technique would be somehow transforming the existing model to a better model using a metamodel and transformational grammar. The linguistic theory plays a vital role to understand the client model and transform it using transformational grammar (Bandler and Grinder, 1975). Therefore, one of the key concerns of psychotherapy is to understand topology of the maladaptive autopilot thoughts and changing the topology of thought process using Mindfulness (Collins et al., 2009), Collaborative Empiricism (Beck and Emery, 1979; Kazantzis et al., 2013) and other talk therapy techniques (Pawelczyk, 2011; Ebert et al., 2015; Mayo-Wilson and Montgomery, 2013; Mohr et al., 2013). In this regards, understanding of topology of uncontrolled dynamic depressive thought (known as *"autopilot thought"* in psychology) is important for evaluating mental health of patients. After a brief discussion on psychological background and motivation behind the work, we will understand how we can represent topology of thought in the next section.

2 How to Represent Topology of Thought

We are considering written text as a symbolic representation of thoughts. To understand the topology of *"autopilot thoughts"*, we have collected tweets of neurotic personality from Twitter applying a hybrid approach combining Deep Learning based classification, KL-Divergence (Manning and Schütze, 1999) based Timeline Similarity Analysis and Rule-based sentiment analysis technique leveraging WordNet-Affect[1] , SentiWordNet[2] and psycholinguistic resource akin to LIWC[3]. Detailed data collection procedure has been discussed in the section 3 and 4 .We are interested to study the representation of words used by neurotic persons in the Vector Space using Word Embedding (Mikolov et al., 2013; Mesnil et al., 2013) and topology (Sizemore et al., 2016) of these semantically embedded words using persistent homology (Zhu, 2013; Kaczynski et al., 2004). We have intuition that timeline of neurotic person contains different topological structure than timeline of user having other personality. Studies say that neurotic person uses more first person pronoun, less social words, more negative emotion words (Pennebaker, 2011). An introvert person uses single topic, discusses more regarding problem, uses few self-references, many tentative words, many negation as compare to extrovert person (Mairesse and Walker, 2007). Topological data analysis using persistent homology has been discussed in the section 5. In the next section, we will discuss our data collection procedure from Twitter.

3 How to Collect Tweets of Neurotic Persons

The proposed approach utilizes an ensemble of state-of-the-art machine learning techniques based on psycholinguistic features to detect distress users (having neurotic personality) from their social media text. We have used Twitter API to search in the Twitter using some seed words/phrases like *"awful"*, *"terrible"*, *"lousy"*, *"hate"*, *"lonely"*, *"hopeless"*, *"helpless"*, *"crap"*, *"sad"*, *"miserable"*, *"tired"*, *"sleep"*, *"hurt"*, *"pain"*, *"kill"*, *"die"*, *"dying"*, *"stressed"*, *"frustrated"*, *"irritated"*, *"depressed"* etc. and name of some antidepression drugs like *"Sertraline"*, *"Citalopram"*, *"Clonazepam"*, *"Propanol"*, *"Prozac"*, *"Zopiclone"*, *"Fluoxetine"*, *"Quetiapine"*, *"Hydroxyzine"* etc. Next, we have filtered out the tweets starting with RT to avoid considering retweets. We have also removed tweets containing url. Thereafter, these tweets are sent to (in house developed) Psychological Annotation Interface for manual annotation. Figure 1 shows screenshot of the interface along with some examples of negative tweets. An annotator can label a tweet considering three aspects Viz:

(a) Personal/Impersonal Emotion Labelling

(b) Polarity Labelling

(c) Psychological Annotation

Individual words in a tweet are annotated according to Psychological Process as discussed

[1] `http://wndomains.fbk.eu/wnaffect.html`

[2] `http://sentiwordnet.isti.cnr.it`

[3] http://liwc.wpengine.com/

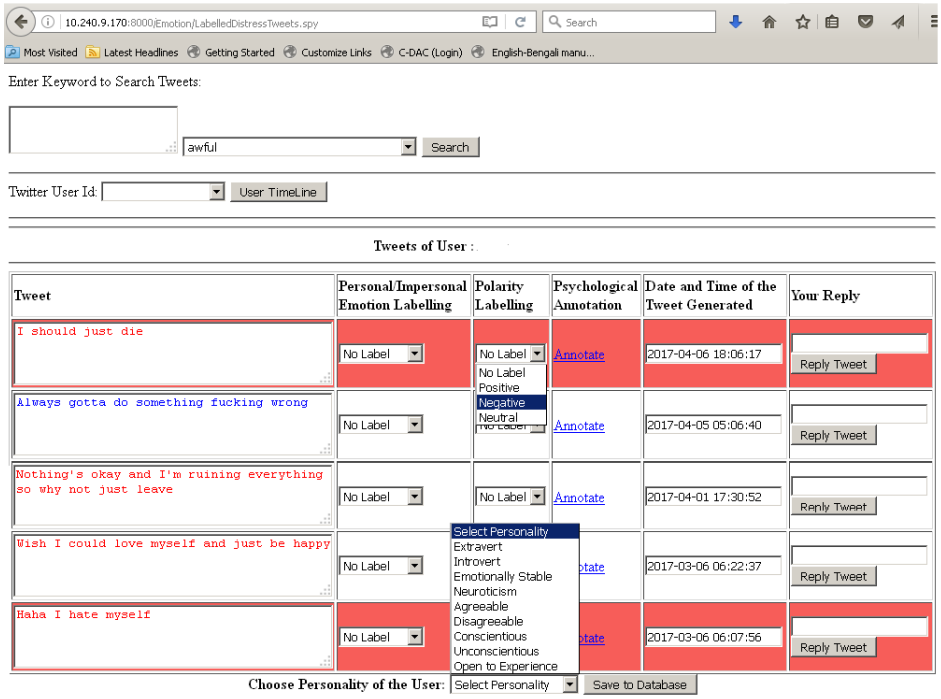


Figure 1: Psychological Annotation Interface

in (Pennebaker et al., 2015). Special care has been taken during annotation to find out "Linguistic Marker of Depression" (Bucci and Freedman, 1981; Pyszcynski and Greenberg, 1987) in the tweet. Pronouns tell us where people focus their attention. If someone uses the pronoun *"I"*, it's a sign of self-focus. Depressed people use the word *"I"* much more often than emotionally stable people (Pennebaker, 2011; Ramirezesparza et al., 2008; Nguyen et al., 2014). Researchers have found that people who frequently use first-person singular words like *"I"*, *"me"* and *"myself"* are more likely to be depressed and have more interpersonal problems than people who often say *"we"* and *"us"*. Using LIWC2001, (Stirman and Pennebaker, 2001) found that suicidal poets were more likely to use first person pronouns (e.g., *"I"*, *"me"*, *"mine"*) and less first plural pronouns (e.g., *"we"*, *"ours"*) throughout their writing careers than were non-suicidal poets. These findings supported the social engagement/disengagement model of depression, which states that suicidal individuals have failed to integrate into society in some way, and are therefore detached from social life (Durkheim, 1951). Similarly, (Rude et al., 2004) found that currently depressed students used more first person singular pronouns, more negative emotional words, and slightly fewer positive emotion words in their essays about coming to college, relative to students who had never experienced a depressive episode. These results are in line with (Pyszcynski and Greenberg, 1987) self-awareness theory. Therefore, in our Psycho-logical Annotation Interface, we have implemented a feature to highlight tweets with red back ground containing First Person Personal Pronoun i.e. *"I"*. Focus on temporal orientation of people that is how often they emphasize the past, present and future is necessary because it affects their health and happiness (Zimbardo and Boyd, 2008). We are interested the proportion of a user's tweets that the analytic finds evidence in: Insomnia and Sleep Disturbance which is often a symptom of mental health disorders (Weissman et al., 1996; De Choudhury et al., 2013), so we have calculated the proportion of tweets that a user makes between midnight and 4 am according to their local time-zone. Therefore, during annotation, special attention should be given on the time perspective of the tweets. Moreover, in the interface we have highlighted tweets in grey colour that have been gener-

ated midnight assuming that the user have sleeping problem or insomnia. From all labelled tweets, unique users name are automatically extracted for analysis of their timelines. Thereafter tweets are extracted from unique users' timeline and annotated using the aforementioned procedure. After annotation of all tweets from a user timeline, the personality of tweet user is labelled in one of the nine classes viz.*Extravert*, *Introvert*, *Emotionally Stable*, *Neuroticism*, *Agreeable*, *Disagreeable*,*Conscientious*,*Unconscientious* and *Open to Experience*. This personality classes are basically extended form of *"Big Five Personality"* (John and Srivastava, 1999) classes. We are more interested for the users those are labelled as *"Neuroticism"* after analyzing the tweets from their timeline. Following the above mentioned procedure, we have created our training data. Manual analysis of tweets collected from users' timeline reveals that information contains in two neurotic users is similar. Moreover, people discuss similar problem among their friend circles, therefore automatic searching *"friend"* and *"followers"* of a neurotic users increases the chance of getting more data of similar nature automatically.
In this way collected training data has been used to train our ensemble learner for detecting depression from social media text in order to collect more tweets from neurotic users. In the next session, we will discuss the depression detection technique using an ensemble classifier.

4 Depression Detection from Social Media Text

4.1 Why Social Media

Currently, depression is primarily assessed through surveys. The standard approach to diagnosing psychological health disorders is through a series of clinically administered diagnostic interviews and tests (Weathers and Davidson, 2001). However, assessment of patients using these tests is expensive and time-consuming. Furthermore, the stigma associated with mental illnesses motivates inaccurate self-reporting by affected individuals and their family members, thus making the tests unreliable. Commonly, the evaluation of a patient is typically performed through the use of standardized questionnaires like Beck Depression Inventory (BDI)[4] , Big Five Inventary (BFI) (John and Srivastava, 1999) etc. A patient's answers are then compiled and compared with disease classification guidelines, such as the International Classification of Diseases or the Diagnostic and Statistical Manual (DSM), to guide the patient's diagnosis. However, these diagnostic methods are not precise and have high rates of false positives and false negatives. In addition, societal and financial barriers prevent many people from seeking medical attention (Michels et al., 2006). Many societies around the world stigmatize and discriminate against people with mental disorders, contributing to the unwillingness of individuals to acknowledge the problem and seek help (Fabrega, 1991). While psychological treatments for depression can be effective (Cuijpers et al., 2008), they are often plagued by access barriers and high rates of attrition (Mohr et al., 2010). Internet interventions have been touted as an antidote to access barriers, but they appear to produce more modest outcomes (Andersson and Cuijpers, 2009), in part also due to high attrition (Christensen et al., 2009). In recent years, there has been a tremendous growth in social interactions on the Internet via social networking sites and online discussion forums. In contrast to clinical tests, the Internet is an ideal, anonymous medium for distressed individuals to relate their experiences, seek knowledge, and reach out for help. Social media is an emerging tool that may assist research in this area, as there exists the possibility of passively surveying and then subsequently influencing large groups of people in real time. (Ruder et al., 2011) have shown that some Facebook users do, in fact, post suicide notes on their profiles, exposing the potential for suicide related research in social media. The amount of publicly available information spread across the realm of social media is extensive. We prefer Twitter because of its greater public availability of data, larger user base, and it being a platform of personal expression. Users generate over 400 million tweets per day (Bennett, 2012). This large reservoir of information regarding

[4] http://www.hr.ucdavis.edu/asap/pdf_files/Beck_Depression_Inventory.pdf

people's daily lives and behaviours, if handled correctly, can be used to study depression, suicide and possibly intervene. Twitter is also used for keeping in touch with friends and colleagues, sharing interesting information within one's network, seeking help and opinions, and releasing emotional stress (Johnston and Hauman, 2013). Therefore, Twitter can be identified as an important surveillance tool for detecting depression and suicidal patterns.

4.2 Methodology

We have applied an ensemble classifier to classify distress and non-distress user based on their social media text collected from Twitter. The ensemble classifier has been built using linear combination of Document Similarity and Emotional Intensity Estimator. Weights in this linear combination are estimated empirically to achieve higher accuracy in this classification task.

We have followed two approaches for detecting depressive tweets viz.

(a) **Document Similarity Measurement:** We have used KL-Divergence (Manning and Schütze, 1999) to measure similarity between searched user's timeline and labelled tweets from all users' timeline and used the similarity score for final scoring of negativity of the user. Applying Latent Dirichlet Allocation (LDA) (Blei et al., 2003), we have estimated topic distribution on the labelled negative[5] and positive data extracted from users' timeline. We have used `sklearn.lda.LDA` library to estimate topic distribution. Then we have estimated the topic similarity of a query user's tweets (user whose personality needs to be estimated) from their timeline with these labelled negative and positive tweets. Then we have estimated the overall similarity score using the following equation:

$$SimilarityScore = 0.6 * \beta + 0.4 * \gamma \quad (1)$$

where β and γ are the similarity scores estimated using LDA and KL-Divergence based topic distribution. Study shows that theme-based retrieval does a better job of finding relevant and effective documents (tweets in user timeline in our case) for this application than conventional approaches (Dinakar et al., 2012b; Dinakar et al., 2012a). All the weights used in the above equations are empirically determined.

(b) **Emotional Intensity Measurement:** We have used following resources for measuring emotional intensity of individual tweet:

(a) SentiWordNet
(b) Manually classified lexicon based on psychological process akin to LIWC.
(c) WordNet Affect

We have calculated *NegetivityScore* combining LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2000; Graves, 2012), a deep learning model for detecting polarity from tweets and rule-based approach using the above mentioned resources and psycholinguistic features.Features used in this classification task are mainly psycholinguistic types, other than that "Pattern of Life Analytics"[6] (Greetham et al., 2011; Berkman et al., 2000; De Choudhury et al., 2013), *"Capitalized Text"*, *"SpecialHashTag"*, *"Probability of Personal Pronoun"*, *"UserName Conatining Special Keywords"* etc. have also been used. Count of some common phrases like *"why me"*, *"I hate myself"* etc. have also been considered as important feature. Examples of "SpecialHashTag" feature are *#depressionprobs*, *#thisiswhatdepressionlike*, *#depression*, *#suicide* etc.. It have been seen that if userid of the users contains some clue substrings like *"depressing"*, *"depression"*, *"hell"*,

[5] In this paper we have used the term "negative tweet" and "distressed tweet" interchangeably to represent the tweet generate by user having neurotic personality.

[6] Social engagement has been correlated with positive mental health outcomes. Tweet rate measures how often a Twitter user posts and pro-portion of tweets with @mentions measures how often a user posts 'in conversation' with other users. Number of @mentions is a measure of how often the user in question engages other users, while Number of self @mentions is a measure of how often the user responds to mentions of themselves.

"depressed", *"sad"*, *"cry"*, *"suicidal"*, *"anxious"*, *"anxiety"*, *"lonely"*, *"die"*, *"broken"*, *"stress"*, *"worthless"*, *"lost"* etc. the timeline of these users contains depressive tweets. Therefore the users having such userid have been considered as an important feature. Tweets written in Upper Case, are considered as important assuming that these are written in Upper Case for providing more importance/intensifying the emotion involved in the tweet.We have used `Theano`, a python based deep leaning library for implementing our LSTM classifier[7].

Final score for selecting neurotic persons has been calculated as follows :

$$FinalScore = \alpha * SimilarityScore + (1 - \alpha) * NegetivityScore \quad (2)$$

Value of alpha ($0 \leq \alpha \leq 1$) can be set experimentally to achieve highest accuracy. We have seen empirically that better result is found when the value of α is 0.8. It has been observed that when the $FinalScore$ is greater than 0.14 then the user can be accepted as neurotic person. Following the procedure discussed in section 3 and 4 we have collected 2500 negative tweets from the timeline of 12 Twitter users having neurotic personality. Same numbers of positive tweets have been collected from the timeline of users having tweets with hashtag *#motivationaltweet*, *#positivethinking*, *#motivationalquotes* etc.

4.3 Vector Space Representation of Distressed Tweets

After manual verification, we have converted the negative and positive tweets into the multidimensional Vector Space. Thereafter, using "t-Distributed Stochastic Neighbor Embedding" (t-SNE) technique (van der Maaten and Hinton, 2008), the higher dimensional data points are projected into a 2d plane. We have used `gensim`[8] python library to convert neurotic persons' tweets into Vector Space. We have considered 2000 dimensions and ±5 context window during Word Embedding. Words that are appeared at least 10 times in the corpus have been selected for vector representation. We have used t-SNE api available in the `sklearn.manifold` library[9] for dimensionality reduction of these higher dimensional points and visualization in the two dimension space. Figure 2 shows representation of embedding words in 2d space using t-SNE. We can see semantically closer words are forming clusters in the Vector Space. *"Kill me"*, *"Suicidal"*, *"destroy"*, *"Cutting"* are appearing closer to each other and *"rejected"*, *"unloved"*, *"worthless"* are forming separate cluster. Separate cluster represent the different topic of the thoughts those are having in the mind of neurotic persons. Conversely, analysing the tweets of positive minded people, we have seen that *"adorable"*, *"comfortable"*, *"eager"*, *"hopeful"*,*"satisfied"* etc. words are frequently used in their timeline. Persistence homology has been applied to the point clouds of positive and negative tweets separately. In the next section we will discuss the topological data analysis of negative and positive tweets based on their vector representation.

5 Topological Data Analysis of Tweets

Persistent homology (Zhu, 2013), a mathematical tool from topological data analysis has been applied on the collected tweets for multiscale analysis on a set of points and identifies clusters, holes, and voids therein. Persistent homology can identify clusters (0-th order holes), holes (1st order, as in our loopy curve), voids (2nd order holes, the inside of a balloon), and so on in a point cloud. It finds "holes" by identifying equivalent cycles. Detailed discussion on Persistent homology[10] and Algebraic Topology is out of scope of the paper. Interested readers can follow work of (Zhu, 2013; Singh et al., 2008; Giblin, 2010; Freedman and Chen, 2011; Zomorodian, 2001; Carlsson, 2008; Edelsbrunner and Harer, 2010; Hatcher, 2002). After representing the words in Vector Space, we have used these data points

[7] `http://deeplearning.net/tutorial/lstm.html`
[8] `https://radimrehurek.com/gensim/`
[9] `http://scikit-learn.org/`
[10] `http://outlace.com/`

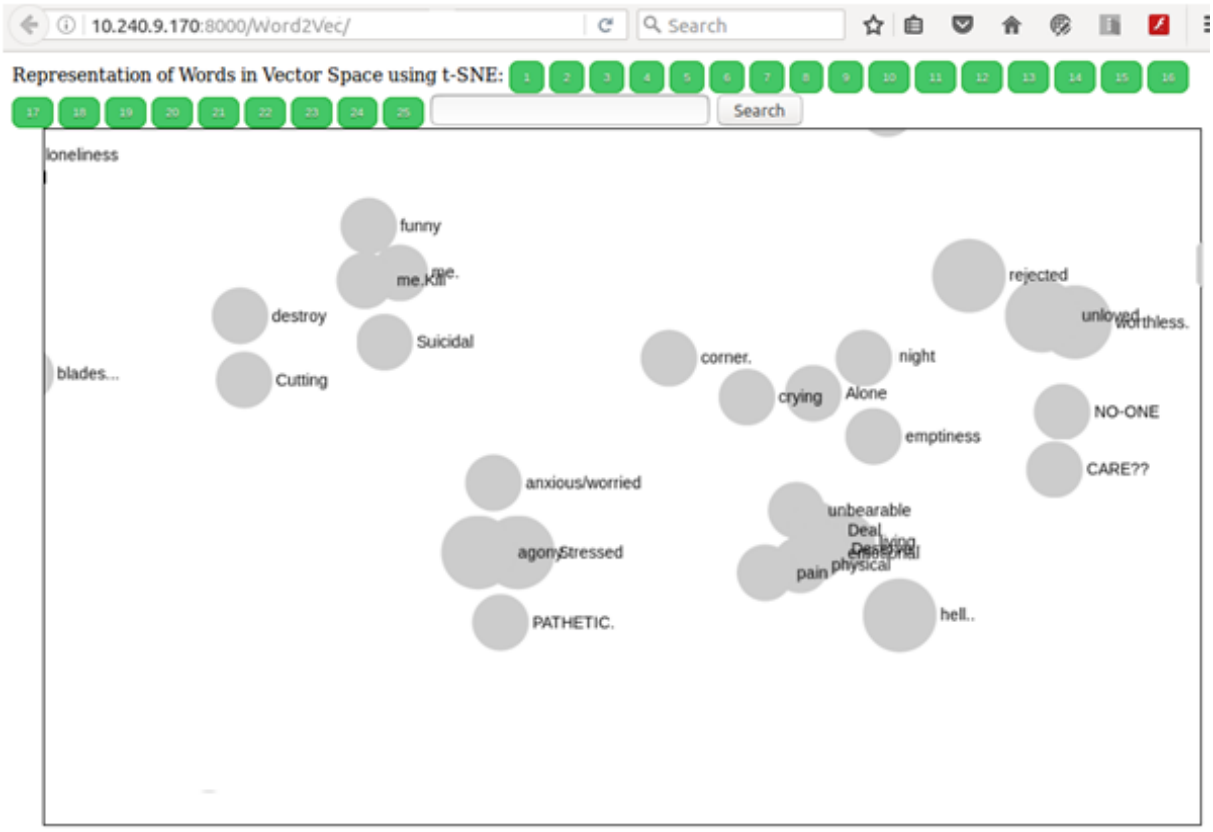


Figure 2: Representation of Words in Vector Space using t-SNE

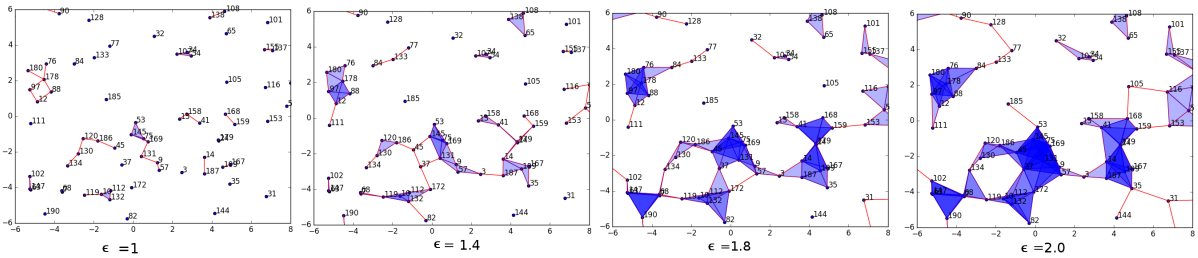


Figure 3: Generated Vietoris-Rips Complexes on Negative Point Cloud with Incremental Values of ϵ.

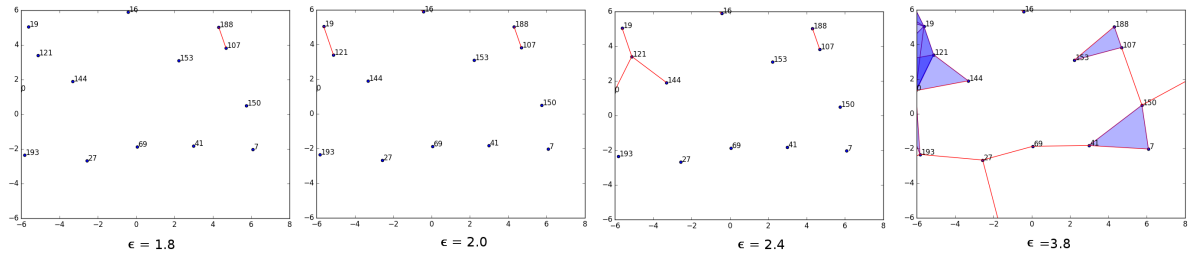


Figure 4: Generated Vietoris-Rips Complexes on Positive Point Cloud with Incremental Values of ϵ.

to build *Vietoris-Rips*[11] complexes of diameter ϵ which are simplicial complexes $VR(\epsilon) = \{\sigma | diam(\sigma) < \epsilon\}$. Here $diam(\sigma)$ represents the largest distance between two points in σ. Distance measures varies according to different contexts. Here we have used euclidean distance for our purpose. Figure 3 and figure 4 show generated *Vietoris-Rips* complexes on negative and positive point cloud respectively. Here we can see, if we set ϵ too small, then generated complexes may just consist of the original point cloud, or only a few edges between the points. If we set ϵ too big, then the point cloud will just become one massive ultradimensional simplex. Our intention is to discover meaningful patterns in a simplicial complex by continuously varying the ϵ parameter (and continually re-build complexes) from 0 to a maximum that results in a single massive simplex. Then we generate a diagram that shows what topological features are born and die as ϵ continuously increases. We assume that features that persist for long intervals over ϵ are meaningful features whereas features that are very short-lived are likely noise. This procedure is called persistent homology computation as it finds the homological features of a topological space that persist while we vary ϵ. Persistent homology examines all ϵ's to see how the system of hole change (also known as *"Birth and Death process"*). An increasing sequence of ϵ produces a filtration. Persistent homology tracks homology classes along the filtration to know for what value of ϵ does a hole appear and how long the hole persists. We have followed the methodology as reported in (Carlsson, 2008) to study the homology of the complexes constructed. The steps involve in this methodology are as follows:

- Construct the $\mathbb{R}$ persistence simplicial complex $\{C_\epsilon\}$ using *Vietoris-Rips* method.
- Select a partial order preserving map $f : \mathbb{N} \to \mathbb{R}$
- Construct the associated $\mathbb{N}$-persistence simplicial complex.
- Construct the associated $\mathbb{N}$-persistence chain complex $\{C_*(n)\}_n$ with co-efficients in F.
- Compute the barcodes associated to the $\mathbb{N}$-persistence F-vector spaces $\{H_i(C_*(n), F)\}_n$

Please refer (Carlsson, 2008) for detail explanation.

The "barcode plot" is a convenient way to visualize persistent homology (Zhu, 2013; Ghrist, 2007). Barcode plot shown in figure 5 and figure 6 are drawn based on increasing sequence of ϵ and zeroth Betti number (β_0) calculated from positive and distressed tweets respectively.

We have selected 500 data points randomly from the word to vector representation of positive and distressed tweets for the filtration process. The word to vector representation using Word Embedding ensures that words that share common contexts (semantics) in the negative and positive tweets are located in close proximity to one another in the Vector Space. Using persistent homology we are trying to examine the topology of these semantically oriented data points (words). The number of connected components is an important topological invariant of a graph. In topological graph theory, it can be interpreted as the zeroth Betti number of the graph. From figure 5, we can see that positive tweets have less connected components (142 disconnected components out of 500 data points) whereas figure 6 shows that negative tweets have much more connected component compare to positive tweets (only 7 disconnected components). Less number of connected components in users' timeline represents wide variation of topics. Conversely, less number of disconnected components indicate that tweets are much more focused towards some specific topics. Manually investigating the contents of the positive and negative tweets, we have seen that users having neurotic personality discuss more regarding their pain and problems. Hence, topic discussed in their timeline more focused to their problem area. On the other hand, positive minded users discuss on different topics and also share ideas and thoughts among their friends and followers. Therefore, tweets generated by them have wide variation of topics. Barcode plot shown in figure 7 and figure 8 are

[11] `http://outlace.com/`

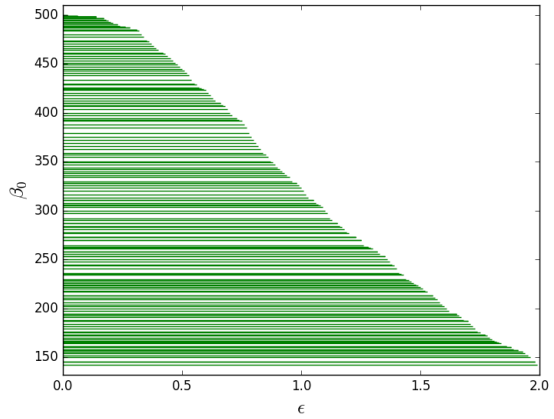


Figure 5: Barcode Plot of Positive Tweets at Betti Dimension 0 (β_0)

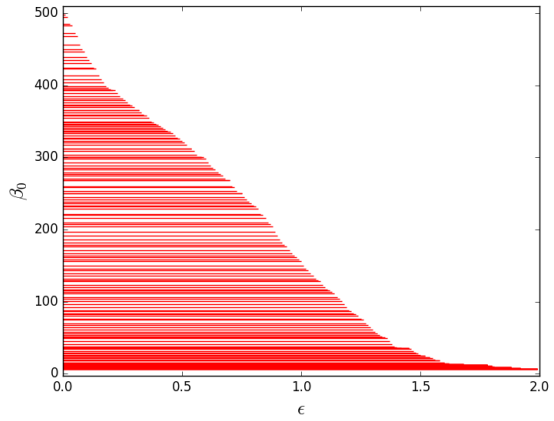


Figure 6: Barcode Plot of Distressed Tweets at Betti Dimension 0 (β_0)

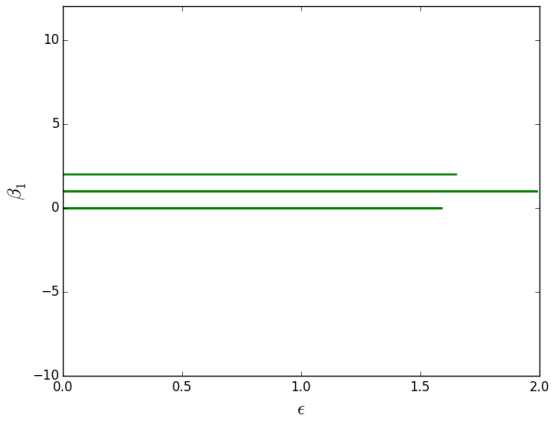


Figure 7: Barcode Plot of Positive Tweets at Betti Dimension 1 (β_1)

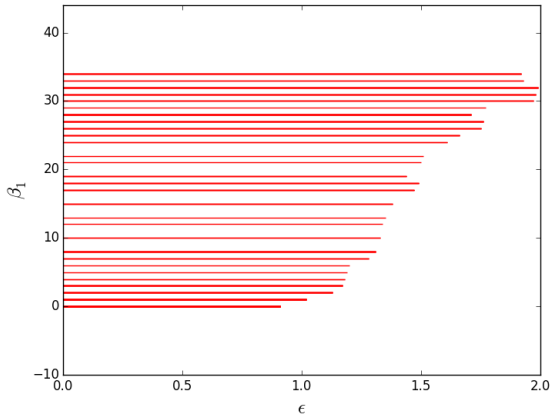


Figure 8: Barcode Plot of Distressed Tweets at Betti Dimension 1 (β_1)

drawn based on increasing sequence of ϵ and 1st Betti number (β_1) calculated from positive and distressed tweets respectively. We can see that figure 7 has very less number of holes and number of holes in the figure 8 are much more compare to figure 7. As the number of disconnected components are much more in the positive tweets, the chance of appearance of one dimensional holes are less. Conversely, number of one dimensional holes are much more in negative tweets because of less number of disconnected components. We have found that number of one dimensional holes in positive tweets is 1 and for negative tweets, it is 34. This observation corroborates the first observations that the people with negative mindset has more oriented set of thoughts (focused to their problem domain) than people having positive mindset. The higher order homology groups produces Betti numbers having values zeros, as expected.

6 Conclusion

In this paper, we have proposed a novel approach for collecting tweets of neurotic persons. Then these tweets are represented in the Vector Space using Word Embedding and dimensionality has been reduced using t-SNE. Persistent homology has been applied to analyse the topology of tweets resembling autopilot thoughts. Psychological features in term of linguistic pattern has been discussed.

As a future work we are planning to explore, how natural language generation can

be applied for therapeutic text generation following RFT and based on topology of patients' thought. We have hypothesized that tweets having psychological features, linguistic markers of depression are indicator of neurotic user's time line as per our understanding of literature. Therefore, as a future work we would like to get expert guidance from psychotherapists for better understanding of the psychological process involved in Mental Health.

The work discussed in this paper is an initiative towards applying NLP in the domain of Mental Health which will motivate researchers for further exploration of linguistic markers and topology involved in psychology.

References

Tim Althoff, Kevin Clark, and Jure Leskovec. 2016. Natural language processing for mental health: Large scale discourse analysis of counseling conversations. *Transactions of the Association for Computational Linguistics*.

G Andersson and P Cuijpers. 2009. Internet-based and other computerized psychological treatments for adult depression: a meta-analysis. *Cognitive Behavioral Therapy*, 38(4):196–205.

R. Bandler and S. Andreas. 1985. *Using Your Brain for a Change*.

R. Bandler and J. Grinder. 1975. *The Structure of Magic I: A Book About Language and Therapy*. Science and Behavior Books.

R. Bandler and J. Grinder. 1979. *Frogs into Princes: Neuro Linguistic Programming*. Real People Press.

Rush A. J. Shaw B. F. Beck, A. T. and G. Emery. 1979. *Cognitive therapy of depression*. Guilford Publications, New york.

Shea Bennett. 2012. Twitter now seeing 400 million tweets per day, increased mobile ad revenue, says ceo@online.

Lisa F. Berkman, Thomas Glass, Ian Brissette, and Teresa E. Seeman. 2000. From social integration to health: Durkheim in the new millennium? *Social Science and Medicine*, 51(6):843—857.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

W. Bucci and N. Freedman. 1981. The language of depression. *Bulletin of the Menninger Clinic*, 45(4):334–358.

Gunnar Carlsson. 2008. Topology and data. Technical report.

H Christensen, KM Griffiths, and Farrer L. 2009. Review adherence in internet interventions for anxiety and depression. *Journal of Medical Internet Research*, 11(2).

SE Collins, N Chawla, SH Hsu, J Grow, JM Otto, and Marlatt GA. 2009. Language-based measures of mindfulness: initial validity and clinical utility. *Society of Psychologists in Addictive Behaviors*, 23(4):743–749.

P Cuijpers, A Van Straten, L Warmerdam, and N Smits. 2008. Characteristics of effective psychological treatments of depression: a meta regression analysis. *Psychother Res*, 18(2):225–36.

Munmun De Choudhury, Scott Counts, and Eric Horvitz. 2013. Predicting postpartum changes in emotion and behavior via social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 3267–3276, New York, NY, USA. ACM.

Karthik Dinakar, Birago Jones, Catherine Havasi, Henry Lieberman, and Rosalind Picard. 2012a. Common sense reasoning for detection, prevention, and mitigation of cyberbullying. *ACM Trans. Interact. Intell. Syst.*, 2(3):18:1–18:30, September.

Karthik Dinakar, Birago Jones, Henry Lieberman, Rosalind W. Picard, Carolyn Penstein Rosé, Matthew Thoman, and Roi Reichart. 2012b. You too?! mixed-initiative LDA story matching to help teens in distress. In *Proceedings of the Sixth International Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012*.

E. Durkheim. 1951. *Suicide*. Free Press, New York.

DD Ebert, AC Zarski, H Christensen, Y Stikkelbroek, P Cuijpers, M Berking, and H Riper. 2015. Internet and computer-based cognitive behavioral therapy for anxiety and depression in youth: a meta-analysis of randomized controlled outcome trials. *PLoS One*, 10(3).

Herbert Edelsbrunner and John Harer. 2010. *Computational Topology - an Introduction*. American Mathematical Society.

Horacio Fabrega. 1991. Psychiatric stigma in nonwestern societies. *Comprehensive Psychiatry*, 32:534–551.

Daniel Freedman and Chao Chen. 2011. Algebraic topology for computer vision. *Computer Vision*, 5:239—268.

Felixv Gers, Jurgen Schmidhuber, and Fred Cummins. 2000. Learning to forget continual prediction with lstm. *Neural Comput*, 12(10):2451–2471.

Robert Ghrist. 2007. Barcodes: The persistent topology of data. Technical report.

Peter Giblin. 2010. *Graphs, Surfaces and Homology*. Cambridge University Press.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer.

David E Greenway, Emily K Sandoz, and David R Perkins. 2010. Potential applications of relational frame theory to natural language systems. In *Proceedings of the Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 2955–2958. IEEE.

Danica Vukadinovic Greetham, Robert Hurling, Gab rielle Osborne, and Alex Linley. 2011. Social networks and positive and negative affect. *Procedia-Social and Behavioral Sciences*, 22:4—13.

Allen Hatcher. 2002. *Algebraic topology*. Cambridge University Press, Cambridge.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

O. P. John and S. Srivastava. 1999. The big-five trait taxonomy: History, measurement, and theoretical perspectives. *Handbook of personality: Theory and research*, 2:102–138.

Chan M.M Johnston, K. and M. Hauman. 2013. Use, perception and attitude of university students towards facebook and twitter. *The Electronic Journal Information Systems Evaluation*, 16(3):201–211, November.

T. Kaczynski, K. Mischaikow, and M. Mozek. 2004. *Computational Homology*. Springer.

Nikolaos Kazantzis, John Tee, Frank Dattilio, and Keith Dobson. 2013. Collaborative empiricism as the central therapeutic relationship element in cbt an expert panel discussion. *International Journal of Cognitive Therapy*, 6(4).

François Mairesse and Marilyn Walker. 2007. Personage: Personality generation for dialogue. In *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 496–503.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.

Evan Mayo-Wilson and Paul Montgomery. 2013. Media-delivered cognitive behavioural therapy and behavioural therapy (self-help) for anxiety disorders in adults. *Cochrane Database Syst Rev*, 9.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In Frédéric Bimbot, Christophe Cerisara, Cécile Fougeron, Guillaume Gravier, Lori Lamel, François Pellegrino, and Pascal Perrier, editors, *INTERSPEECH*, pages 3771–3775. ISCA.

Kathleen M. Michels, Karen J. Hofman, Gerald T. Keusch, and Sharon H. Hrynhow. 2006. Stigma and global health: Looking forward. *Lancet*, 367:538–539.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Adam Miner, Amanda Chow, Sarah Adler, Ilia Zaitsev, Paul Tero, Alison Darcy, and Andreas Paepcke. 2016. Conversational agents and mental health: Theory-informed assessment of language and affect. In *Proceedings of the Fourth International Conference on Human Agent Interaction*, HAI '16, pages 123–130, New York, NY, USA. ACM.

DC Mohr, J Ho, J Duffecy, KG Baron, KA Lehman, L Jin, and D Reifler. 2010. Perceived barriers to psychological treatments and their relationship to depression. *Journal of Clinical Psychology*, 66(4):394–409.

David C Mohr, Michelle Nicole Burns, Stephen M Schueller, Gregory Clarke, and Michael Klinkman. 2013. Behavioral intervention technologies: evidence review and recommendations for future research in mental health. *General hospital psychiatry*, 35(4):332–338.

Thin Nguyen, Dinh Phung, Bo Dao, Svetha Venkatesh, and Michael Berk. 2014. Affective and content analysis of online depression communities. *IEEE Transactions on Affective Computing*, 5(3).

Joanna Pawelczyk. 2011. *Talk as Therapy: Psychotherapy in a Linguistic Perspective*. Walter de Gruyter.

James W. Pennebaker, Matthias R. Mehl, and Kate G. Niederhoffer. 2003. Psychological aspects of natural language use: Our words, our selves. *Annual review of psychology*, 54(1):547–577.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of liwc2015. *UT Faculty/Researcher Works*.

James W. Pennebaker. 2011. *The secret life of pronouns: What our words say about us*. Bloomsbury Press, New York.

T. Pyszcynski and J. Greenberg. 1987. Self-regulatory perseveration and the depressive self-focusing style: A self-awareness theory of depression. *Psychological Bulletin*, 102:122–138.

Nairan Ramirezesparza, Cindy K. Chung, Ewa Kacewicz, and James W. Pennebaker. 2008. The psychology of word use in depression forums in english and in spanish: Testing two text analytic approaches. In *Proceedings of the ICWSM*.

Stephanie Rude, Eva-Maria Gortner, and James Pennebaker. 2004. Language use of depressed and depression-vulnerable college students. *Cognition and Emotion*, 18:1121–1133.

Thomas Ruder, Gary M Hatch, Garyfalia Ampanozi, and Nadja Fischer. 2011. Suicide announcement on facebook. *Crisis*, 32(5):280–282.

Gurjeet Singh, Facundo Memoli, Tigran Ishkhanov, Guillermo Sapiro, Gunnar Carlsson, and Dario L. Ringach. 2008. Topological analysis of population activity in visual cortex. *Journal of Vision*, 8(8):1–18.

Ann Sizemore, Chad Giusti, Ari Kahn, Richard F. Betzel, and Danielle S. Bassett. 2016. Cliques and cavities in the human connectome. *arXiv:1608.03520*.

S. W. Stirman and J. W. Pennebaker. 2001. Word use in the poetry of suicidal and non-suicidal poets. *Psychosomatic Medicine*, 63:517–522.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, November.

M Villatte, JL Villatte, and SC Hayes. 2015. *Mastering the Clinical Conversation: Language as Intervention*. Guilford Publications, Palo Alto, CA, October.

Keane T. M. Weathers, F. W. and J. Davidson. 2001. Clinician-administered ptsd scale: A review of the first ten years of research. *Depression and Anxiety*, 13:132–156.

Myrna M. Weissman, Roger C. Bland, Glorisa J. Can-ino, Carlo Faravelli, Steven Greenwald, HaiGwo Hwu, Peter R. Joyce, Eile G. Karam, Chung-Kyoon Lee, Joseph Lellouch, Jean-Pierre Lepine, Stephen C. Newman, Maritza Rubio-Stipec, J. Elisabeth Wells, Priya J. Wickramaratne, Hans-Ulrich Wittchen, and Eng-Kung Yeh. 1996. Cross-national epidemiology of major depression and bi-polar disorder. *Journal of the American Medical Association (JAMA)*, 276(4):293—299.

Xiaojin Zhu. 2013. Persistent homology: An introduction and a new text representation for natural language processing. In Francesca Rossi, editor, *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 1953–1959. IJCAI/AAAI.

P.G. Zimbardo and J.N. Boyd. 2008. *The Time Paradox: The new psychology of time that will change your life*. Free Press, New York.

Afra Joze Zomorodian. 2001. *Computing and comprehending topology: persistence and hierarchical Morse complexes*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

A Deep Dive into Identification of Characters from Mahabharata

Apurba Paul and Dipankar Das

Dept. of Computer Science and Engineering,
Jadavpur University,Kolkata,West Bengal,India

Abstract

The present paper describes the identification of story Characters from Indian Mythological text "The Mahabharata". It is observed that these Characters can be found at word level and phrase level in a sentence with some distinct patterns. In order to find the Characters from the text, two sets of features are considered at both levels. Using a semi-supervised learning approach we have prepared the training data sets. Later on, we have employed Chi-squared statistic to find the important features, which is followed by the associativity analysis of those selected features. After that, we developed training models using NeuralNet and KNN classifiers for both word and phrase levels and tested the models. Our observation shows that NeuralNet performs better than KNN with 88% and 76% accuracy at word and phrase level respectively.Next, we have analyzed different error measures followed by visualization of co-occurred story Characters of most frequent Characters.

1 Introduction

Characters has a significant role and takes part in several activities throughout any stories. They may or may not be lovable, respectable, honorable, graceful, disgraceful, cruel, selfish but the readers do need to understand them and why they act the way they do in the texts. A character, being a protagonist, is commonly on the good side while the antagonist is the one he/she fights or has conflicts within the story. In the stories, Characters may have dialogues, actions which influence the plot of the texts, emotions etc. They respond to events and other characters through what they say or don't say, what they do and don't do, what they think, and what they feel. Character's thoughts in response to the actions or words of others are obviously a key to that Character's personality. Like thoughts, Characters emotions can instantly reveal a Character's personality and what he/she finds important. If we dive deep in the story we can extract the actions, thoughts, emotions and overall personality of a Character easily. Since the Characters are playing the major role in any story, we can consider automatic identification of Characters from stories is one of the primary task .

In the similar context, we can find several Characters in the Indian epic "The Mahabharata". Here the Characters may be protagonist or antagonist. So the extraction and identification of Characters are very important. In this text we can find that a Character may appear in word level(NNP) or it may appear in a phrase level(NP<<NNP). As an example, "Yudhisthira"(NNP) is a Character at word level and simultaneously "The Kuru king Yudhishthira" (NP<<NNP) is also a Character at phrase level. But it is also seen that only NNP or NP<<NNP are not sufficient rule to identify a Character in the texts. So the identification of Characters at word and phrase level is the main research issue addressed over here.

In this paper, we have employed two different approaches to address the presence of a Character in Mahabharata. We have identified at word level a set of 97 features and at phrase level a set of 51 features. With the help of these features we have developed our data sets and later on devised two different training models using semi supervised approach. Next, we identified the set of important features and their associativity among them. After that we have tested our model and observed the precision, recall, f-measure, kappa and errors.

In the rest of the paper, we have discussed related work and the data preparation steps followed by

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 447–455

experiments, result and error analysis, visualization of co-occurred Characters and conclusion.

2 Related Work

There are a few works done on Character Identification from texts. Paul and Das (2017) proposed a rule based system by which they can extract the Character Adjectives from the Indian mythological text Mahabharata. Valls-Vargas et al. (2015) also proposed a feedback-loop-based approach to identify the characters and their narrative roles where the output of later modules of the pipeline is fed back to earlier ones. Valls-Vargas et al. (2014) proposed a case-based approach to character identification in natural language text in the context of their Voz system. Valls-Vargas et al. (2013) proposed a method for automatically assigning narrative roles to characters in stories. Calix et al. (2013) developed a methodology to detect sentient actors in the spoken stories. Goyal et al. (2010) proposed a system that exploits a variety of existing resources to identify affect states and applies to map the affect states onto the characters in a story. Mamede and Chaleira (2004) developed a system (DID) which was applied to children stories starts by classifying the utterances. The utterances belong to the narrator (indirect discourse) as well as belong to the characters taking part in the story (direct discourse). Afterwards, this DID system tries to associate each direct discourse utterance with the character(s) in the story. In the context of keyword extraction, statistical approaches are often built for extracting general terms (Van Eck et al., 2010).

3 Data Preparation

In this paper we consider Mahabharata as a case study from where we choose aswamedha, asramvasika, mausala, mahaprasthanika and svargarohanika parva(or Chapter) as our sample space. We can observe that there exists a lot of Characters which plays a significant role in these texts. At first we annotate these Characters manually and made a list of Characters out of it . Then to understand the positions and occurrences of each Characters we investigate each sentences in the texts with the help of Stanford CoreNLP suite. We tokenized each sentences, annotate them with POS tagger and generate syntactic parse tree by the suite. After a detail observation of each sentence in each text we developed a notion that Characters can be found in word level and phrase level as well. We also observed that in most of the cases at word level, a word is a Characters when its POS tag is NNP. Similarly at phrase level, a phrase is a Character when the root of the phrase is NP and one of its descendant is NNP. The examples are given below.

At word Level:
(NNP Narayana)=[Narayana]$_{\text{Character}}$
At phrase Level:
(NP (DT the) (JJ holy) (NNP Rishi) (NNP Vyasa)) = [The holy Rishi Vyasa]$_{\text{Character}}$

3.1 Feature set Generation

The above observation helps us to extract different features at word level and phrase level. The list of features at both the levels with appropriate examples are explained in the next sub section.

3.1.1 Word Level Features

For each NNP present in a sentence at word level we have considered 97 different features. They are displayed in Table 1:

Word Level Features(WL_F)		
Sl(W)	**Name**	**Freq.**
1	Extracted NNP word(Cw)	4152
2	NNP-tag	4152
3	Length of Cw	4152
4	Starting Index of Cw	4152
5	Ending Index of Cw	4152
6	Previous word of Cw	3583
7	Previous word tag of Cw	2584
8	Next word of Cw	4152
9	Next word tag of Cw	4152
10	Porter Stemmer word of Cw	4152
11	Is porter Stemmed word same with Cw?	4152
12	Snowball Stemmer word of Cw	4152
13	Is snowball stemmed word same with Cw?	4152
Immediate Pre and Post . . . Features of Cw		
14-17	verb word and tag	2645,3158
18-21	adverb word and tag	1218,1381
22-25	preposition word and tag	2590,3042
26-29	noun word and tag	2741,3412

Continued on next page

Continued from previous page		
Sl(W)	**Name**	**Freq.**
30-33	NNP word and tag	2199,2229
34-37	adjective word and tag	1445,2089
38-41	C. Conjunc. word and tag	1052,1611
42-45	determiner word and tag	2608,2554
46-49	existential word and tag	0092,0043
50-53	interjection word and tag	0001,0001
54-57	TO word and tag	0541,0959
58-61	Cardinal Number and tag	0255,0263
62-65	pronoun word and tag	1162,1773
66-69	Wh word and tag	0625,0771
Immediate Pre and Post ... Distance from Cw		
70,71	verb distance	2645,3158
72,73	adverb distance	1218,1381
74,75	preposition distance	2590,3042
76,77	noun distance	2741,3412
78,79	NNP distance	2199,2229
80,81	adjective distance	1445,2089
82,83	C Conjunction distance	1052,1611
84,85	determiner distance	2608,2554
86,87	existential distance	0092,0043
88,89	interjection distance	0001,0001
90,91	TO distance	0541,0959
92,93	Cardinal Number distance	0255,0263
94,95	pronoun distance	1162,1773
96,97	Wh distance	0625,0771
		Concluded

Table 1: List of Word Level Features(WL_F)

In the Table 1, mainly we have categorized the set of features in three different sub categories.The features from W1 to W13 are sub categorized as general features of a context word(Cw), from W14 to W69, the features are sub categorized as Immediate pre and post word and tag of Cw and from W70 to W97, the features are responsible for counting the word distance from the context word Cw as immediate pre and post word distance, along with their frequencies.Here frequency reveals the number of occurrences of a distinct feature in our sample space. As an example, consider a feature set W14-17(verb word and tag). It contains four different types of features.The W14 is immediate pre verb word which is situated in the left of Cw and W15 is its POS Tag with frequency 2645. Next, W16 is immediate post verb word situated in the right side of Cw in a sentence and W17 identifies its POS Tag with frequency 3158. Again as an example consider W70,71(verb distance). Here W70 calculates the word distance of verb situated in the left of Cw as immediate pre verb distance.The frequency of this feature is 2645. Likewise, the W71 finds the word distance of verb situated in the right of Cw as immediate post verb with frequency 3158.

Consider a sentence $\mathbf{S_1}$= *"Having bowed down unto Narayana, and to Nara, the foremost of men, as also to the goddess Sarasvati, should the word Jaya be uttered."*

In the above sentence our context word(Cw) is $\mathbf{Narayana_{Character}}$. Some of the features extracted from the sentence $\mathbf{S_1}$ with respect to $\mathbf{Narayana_{Character}}$ are explained in Figure 1.

3.1.2 Phrase Level Features

At phrase level, we have considered 51 different features displayed in Table 2 for each NP<<NNP pattern present in the sentences.

Phrase Level Features(PL_F)		
Sl(P)	**Name**	**Freq.**
1	Current head Node of the phrase(Ch)	2991
2	The pre terminal yield Nodes of Ch	2991
3	Leaves of the Ch(Cw)	2991
4	Path from Ch to ancestor Node	2991
5	has ADJP as siblings of Ch?	0018
6	has ADJP as siblings of Ch?	0128
7	has CONJP as siblings of Ch?	0006
8	has FRAG as siblings of Ch?	0001
9	has INTJ as siblings of Ch?	0001
10	has LST as siblings of Ch?	0001
11	has NAC as siblings of Ch?	0001
12	has NP as siblings of Ch?	0790
13	has NX as siblings of Ch?	0001
14	has PP as siblings of Ch?	0271
15	has PRN as siblings of Ch?	0016
16	has PRT as siblings of Ch?	0001
17	has QP as siblings of Ch?	0001
18	has RRC as siblings of Ch?	0003
19	has UCP as siblings of Ch?	0003
20	has VP as siblings of Ch?	0619
21	has WHADJP as siblings of Ch?	0001
		Continued...

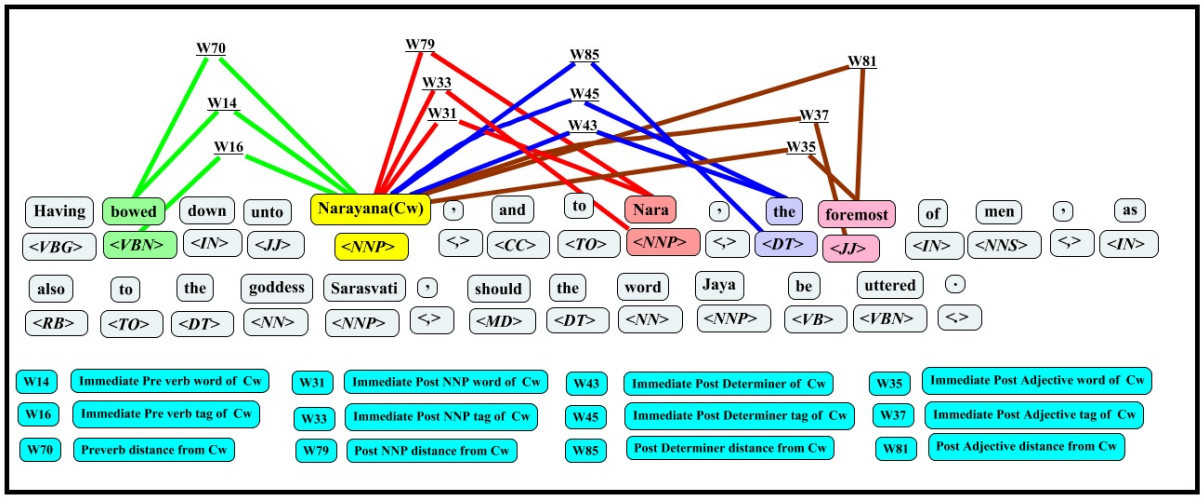


Figure 1: Example of Word Level Features

Continued...		
Sl(P)	**Name**	**Freq.**
22	has WHAVP as siblings of Ch?	0001
23	has WHNP as siblings of Ch?	0001
24	has WHPP as siblings of Ch?	0001
25	has X as siblings of Ch?	0004
26	has COMMA as siblings of Ch?	0788
27	has STOP as siblings of Ch?	0313
28	Ancestor Node of Ch(AnCh)	2991
29	has ADJP as siblings of AnCh?	0008
30	has ADJP as siblings of AnCh?	0088
31	has CONJP as siblings of AnCh?	0028
32	has FRAG as siblings of AnCh?	0001
33	has INTJ as siblings of AnCh?	0001
34	has LST as siblings of AnCh?	0001
35	has NAC as siblings of AnCh?	0001
36	has NP as siblings of AnCh?	1239
37	has NX as siblings of AnCh?	0001
38	has PP as siblings of AnCh?	0210
39	has PRN as siblings of AnCh?	0016
40	has PRT as siblings of AnCh?	0018
41	has QP as siblings of AnCh?	0001
42	has RRC as siblings of AnCh?	0001
		Continued...

Continued...		
Sl(P)	**Name**	**Freq.**
43	has UCP as siblings of AnCh?	0001
44	has VP as siblings of AnCh?	0470
45	has WHADJP as siblings of AnCh?	0001
46	has WHAVP as siblings of AnCh?	0001
47	has WHNP as siblings of AnCh?	0025
48	has WHPP as siblings of AnCh?	0001
49	has X as siblings of AnCh?	0001
50	has COMMA as siblings of AnCh?	0706
51	has STOP as siblings of AnCh?	0446
		Concluded

Table 2: List of Phrase Level Features(PL_F)

In the Table 2 it can be observed that there are mainly two different subcategories of features. All the features from P1 to P27 are related to the phrase(Cw) which is assumed to be a Character, and rest of the features are related to the two level up ancestor(parent of a parent of Current head node,Anch), along with their frequencies. Here frequency identifies the number of occurrences of a particular feature in the sample space. As an example, P1 contains the Current head Node of the phrase(Ch) with frequency 2991 and P36 finds the existence of any NP as a sibling of Ancestor Node of Ch(AnCh). The frequency of P36 is 1239.

Again Consider a sentence $\mathbf{S_2}$= *"The king, in honour of Hari and naming him repeatedly, fed* ***the Island-born Vyasa****, and Narada, and Markandeya possessed of wealth of penances, and Yajnavalkya of Bharadwaja's race, with many delicious viands."*

The important part of the parse tree of the above sentence $\mathbf{S_2}$ is ,

$\mathbf{S_{2parsed}}$= (VP (VBN fed) (NP (NP (DT the) (JJ Island-born) (NNP Vyasa)) (, ,) (CC and) (NP (NNP Narada))))

In the above sentence our target phrase is **the Island-born Vyasa$_{\mathbf{Character}}$**. Figure 2 explains the features P1,P2,P3 and P28 in details.

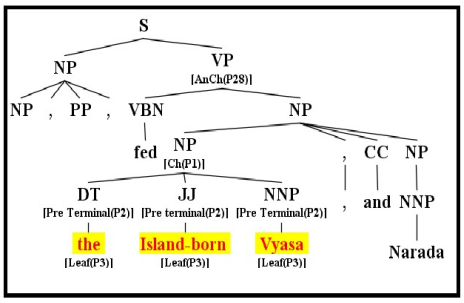


Figure 2: Example of Phrase Level Features P1,P2,P3,P28

3.2 Training & Test set Preparation

To prepare the training sets for both word level and phrase level we consider semi supervised learning approach. At first, we have extracted all the features of each NNP present in mahaprasthanika parva at word level and compare each NNP with manually tagged list of Characters. The NNP's which are found in the list are annotated as Character and in case of unavailability they are termed as Not_Character. In this way we have prepared a data set, $WD_{training}$. Next, we have extracted all the features of each NNP present in svargarohanika parva and prepared a dataset called WD_{test} . After that we developed a learning model trained on $WD_{training}$ data set using KNN-classifier and test the model using WD_{test} . Then we calculate the accuracy, precision, recall and f-measure of WD_{test}. Later on all the NNP's in WD_{test} dataset are annotated properly and update the $WD_{training}$ dataset by appending newly an-notated WD_{test} dataset. This process is repeated for all other chapters in our sample space. Finally we got an updated dataset $WD_{training}$ which is considered as a training set containing word level features for our system. The results are discussed in Table 3.

Word Level ($WD_{Training}$)			
Parva	**P**	**R**	**F**
svargarohanika	0.43	0.44	0.43
mausala	0.62	0.64	0.60
asramvasika	0.63	0.63	0.63
aswamedha	0.71	0.69	0.68
P=Precision; R=Recall; F=F-measure			

Table 3: Precision, Recall, F-measure at Word Level

Similarly, at phrase level we have extracted all the features of each NP<<NNP present in the mahaprasthanika parva and prepared a data set named as $PD_{training}$ and trained a model with KNN Classifier. Next, we have extracted all the features of each NP<<NNP present in svargarohanika parva and prepared a dataset called PD_{test} which is applied on the trained model like word level process. Here, we calculate precision, recall and f-measure of PD_{test} . This process is iterated for other chapters and finally we got updated $PD_{training}$ as a training set of phrase level. The results are observed in Table 4.

Phrase Level ($PD_{Training}$)			
Parva	**P**	**R**	**F**
svargarohanika	0.52	0.56	0.50
mausala	0.67	0.65	0.63
asramvasika	0.60	0.62	0.60
aswamedha	0.58	0.52	0.44
P=Precision; R=Recall; F=F-measure			

Table 4: Precision, Recall, F-measure on Phrase Level

At last we choose virata parva as a test case, annotate all the Characters present in the text and made a list out of it. Next, we prepared the data sets for word and phrase level, WT_{Test} and PT_{Test}respectively. Then we have mapped all the NNP, NP<<NNP present in the virata parva with Character and Not_Character which we will refer to in the Result Analysis section.

4 Experiments

To find the important features in $WD_{Training}$ and $PD_{Training}$ datasets we calculated the relevance of the features by computing the Chi squared statistic with respect to the Class level feature

using Rapid Miner tool[1]. The higher the weight of a feature, the more relevant it is considered. The value of the Chi Squared statistic is given by

$$X^2 = \sum \frac{(O-E)^2}{E} \quad (1)$$

where, X^2 is the chi-square statistic, O is the observed frequency and E is the expected frequency. Using this measure we got 46 important feature at word level and 9 at phrase level. The list of relevant features derived from the measure at word level(D_w) is given in Table 5.

Word Level Features(D_w)	
Sl	**Name**
W1	Extracted NNP-word(Cw)
W4	Starting Index of Cw
W5	Ending Index of Cw
W6	Previous word of Cw
W7	Previous word tag of Cw
W8	Next word of Cw
W9	Next word tag of Cw
W10	Porter Stemmer word of Cw
W11	Is porter Stemmed word same with Cw?
W12	Snowball Stemmer word of Cw
W13	Is snowball stemmed word same with Cw?
Immediate Pre and Post ... Features of Cw	
W14	Pre verb word
W15	Pre verb tag
W16	Post verb word
W20	Post adverb word
W22	Pre preposition word
W23	Pre preposition tag
W26	Pre noun word
W27	Pre noun tag
W28	Post noun word
W29	Post noun tag
W30	Pre NNP word
W31	Pre NNP tag
W32	Post NNP word
W35	Pre adjective tag
W38	Pre C. Conjunction word
W39	Pre C. Conjunction tag
W64	Post pronoun word
W68	Post Wh word
W69	Post Wh tag
Immediate Pre and Post ... Distance from Cw	
W70	Pre verb distance
W71	Post verb distance
W73	Post adverb distance
W74	Pre preposition distance
W75	Post preposition distance
W76	Pre noun distance
W77	Post noun distance
W78	Pre NNP distance
W80	Pre adjective distance
W82	Pre C Conjunction distance
W83	Post C Conjunction distance
W84	Pre determiner distance
W85	Post determiner distance
W95	Post pronoun distance
W96	Pre Wh distance
W97	Post Wh distance

Table 5: List of Relevant Features at Word Level (D_w)

Next, the list of relevant features at phrase level extracted by the above method is described in Table 6.

Phrase Level Features(D_p)	
Sl	**Name**
P1	Current head Node of the phrase(Ch)
P3	Leaves of the Ch(Cw)
P4	Path from Ch to ancestor Node
P12	has NP as siblings of Ch?
P20	has VP as siblings of Ch?
P26	has COMMA as siblings of Ch?
P27	has STOP as siblings of Ch?
P28	Ancestor Node of Ch(AnCh)
P36	has NP as siblings of AnCh?

Table 6: List of Relevant Features at Phrase Level (D_p)

Now with the help of D_w and D_p we prepared our new training sets as D_{wt} and D_{pt} .Similarly we have prepared our new test sets with these important features as D_{wtest} and D_{ptest} from the text virata parva .

4.1 Features Associativity Analysis

It is observed from the training data sets, D_{wt} and D_{pt}, that some feature or set of features coexists

[1]https://rapidminer.com

with other feature or set of features. This type of relations can be found from the texts very frequently in our sample space. To address this issue we have applied FP-Growth algorithm in word and phrase level. This algorithm calculates all frequent feature/feature set from the data set by building a FP-Tree data structure on the data sets D_{wt} and D_{pt} . Some frequent relations of word and phrase level are given below.

Word Level relations:

Word Level		
Antecedent	**Consequent**	**Confidence**
W20	W14	0.258
W20	W83,W35	0.408
W23	W83,W31,W27	0.352
W20,W35	W14	0.381
W20,W83	W35,W31	0.381
Antecedent -> Consequent		
W14=Immediate pre verb word of Cw W20=Immediate post adverb word of Cw W23=Immediate pre position tag of Cw W27= Immediate pre noun tag of Cw W31=Immediate pre NNP tag of Cw W35=Immediate pre adjective tag of Cw W83=Post C. Conjunction distance from Cw		

Table 7: Features Associativity at Word Level

From Table 7 we can understand that for a distinct context word, Cw,when we identify a value for the feature,W20 in a sentence in the sample space, simultaneously we can find a value for the feature W14 also with a confidence value 0.258.

Phrase Level relations:

Phrase Level		
Antecedent	**Consequent**	**Confidence**
P3	P26	0.261
P26	P3	0.412
P3	P12	0.418
P12	P3	0.743
P26	P12	0.659
P12	P26	0.795
Antecedent -> Consequent		
P3 = Leaves of the Ch(Cw) P12= hasNP as siblings of Ch? P26= hasCOMMA as siblings of Ch?		

Table 8: Features Associativity at Phrase Level

Similarly in the Table 8, when we can observe a value for the feature P3 then P26 is also observed for context word Cw in a sentence of our sample space with confidence value 0.261.

Where X–> Y implies that if X occurred then Y also occurred; X means antecedent and Y means consequent.

4.2 Classification Task

Here we have developed a training model using NeuralNet and KNN classifiers with the help of newly prepared datasets D_{wt} and D_{pt} .Later on we have tested these models using our newly created test sets D_{wtest} and D_{ptest} .At word level NeuralNet has better precision, recall and f-measure than KNN classifier. At phrase level NeuralNet classifier has better precision and f-measure than KNN classifier. On the other hand KNN has better recall value than NeuralNet classifier at phrase level. The precision, recall and f-measure of the two classifiers are explained in Table 9 and Table 10.

Word Level			
Classifiers	**P**	**R**	**F**
NeuralNet	91.84	84.91	88.24
KNN	90.70	73.58	81.25
P=Precision; R=Recall; F=F-measure			

Table 9: Precision, Recall, F-measure on D_{wtest}

Phrase Level			
Classifiers	**P**	**R**	**F**
NeuralNet	79.07	69.39	73.91
KNN	61.67	75.51	67.89
P=Precision; R=Recall; F=F-measure			

Table 10: Precision, Recall, F-measure on D_{ptest}

5 Result Analysis

Both the classifiers performed well in case of classifying the test data sets D_{wtest} and D_{ptest}. NeuralNet classifier has better accuracy than KNN classifier in case of word level and phrase level. The accuracies of both the classifiers for word level and phrase level are discussed in Table 11.

Classifiers	**WAccuracy**	**PAccuracy**
NeuralNet	88.00%	76.00%
KNN	82.00%	65.00%
WAccuracy=Word Level Accuracy		
PAccuracy=Phrase Level Accuracy		

Table 11: Classification Accuracies

The confusion table on accuracies of D_{wtest} and D_{ptest} are given in Table 12 and Table 13 below.

Word Level				
Classifiers	**NN**	**NC**	**CN**	**CC**
NeuralNet	45	4	8	43
KNN	39	4	14	43
C=Character;N=Not_Character				

Table 12: Confusion table for dataset D_{wtest}

From Table 12 we can observe that NeuralNet classifier has correctly classified 88 instances and incorrectly classified 12 instances.Whereas KNN classifier has classified 82 instances correctly and 18 instances incorrectly.

Phrase Level				
Classifiers	**NN**	**NC**	**CN**	**CC**
NeuralNet	42	15	9	34
KNN	28	12	23	37
C=Character;N=Not_Character				

Table 13: Confusion table for dataset D_{ptest}

On the other hand in Table 13 at Phrase level, NeuralNet classifier has classified 24 instances incorrectly and 76 instances correctly.Similarly KNN classifier has classified 35 instances incorrectly and 65 instances correctly.

6 Error Analysis & Observations

It can be observed that NeuralNet classifier has lowest classification error and highest kappa value at word level and phrase level as well. The classification error rate and kappa measure are observed in Table 14.

Classifiers	**Word Level**		**Phrase Level**	
	CE	**K**	**CE**	**K**
NeuralNet	12.00%	0.760	24.00%	0.519
KNN	18.00%	0.643	35.00%	0.303
CE=Classification Error rate; K=Kappa measure				

Table 14: Error and Kappa of D_{wtest} and D_{ptest}

The average absolute deviation of the prediction from the actual value, i.e., Absolute Error of NeuralNet classifier at word level is lower than KNN classifier. Similarly the average of the absolute deviation of the prediction from the actual value divided by the actual value, i.e., Relative Error, and Root Mean Squared Error of NeuralNet classifier at word level significantly lower than KNN classifier. The details are explained in Table 15.

Word Level		
Measures	**NeuralNet**	**KNN**
AE	0.16+/-0.28	0.31+/-0.19
RE	16.83+/-28.09	31.83+/-18.97
RMSE	0.32+/-0.00	0.37+/-0.00
AE=Absolute Error; RE=Relative Error(%)		
RMSE= Root mean Squared Error		

Table 15: Error Analysis of D_{wtest}

At phrase level also NeuralNet classifier has lower Absolute Error, Relative Error and Root Mean Squared Error than KNN classifier. The results are given in Table 16.

Phrase Level		
Measures	**NeuralNet**	**KNN**
AE	0.36+/-0.24	0.41+/-0.25
RE	36.93+/-23.97	41.89+/-25.84
RMSE	0.44+/-0.00	0.492+/-0.00
AE=Absolute Error; RE=Relative Error(%)		
RMSE= Root mean Squared Error		

Table 16: Error Analysis of D_{ptest}

7 Visualization of Co-Occurred Characters

Now, we have measured the co occurrence of all the Characters extracted at word level and phrase level. We analyzed each sentence in our sample space and calculated the co occurrence of each Characters with others. As an example we considered a Character found at word level C="Abhimanyu". In the Figure 3 we have displayed the list of co occurred Characters related to Character C.

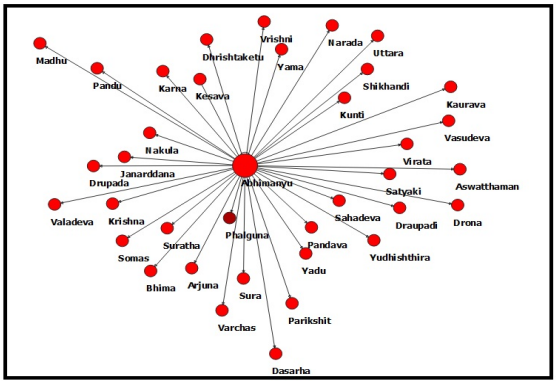


Figure 3: Co occurred Characters w.r.t Character Abhimanyu

8 Conclusion

In this paper our target is to identify the Characters from the Mahabharata. Keeping this in mind at first we have annotated all the Characters present in the sample space. Then we have applied a semi supervised approach with distinct features at word level and phrase level to collect the Characters from the texts and prepared the initial data sets. Then we applied Chi Squares Statistic to find the relevant features from the data sets. According to the relevant features at word and Phrase level we have reshaped our training and testing data sets. Next, we have analyzed the associativity of features using FP-Growth algorithm. Here we found that some features has coexistence with other feature or set of features. Then we developed training models at word level and phrase level with NeuralNet and KNN classifier. Later we have tested our model with our test data and accuracies, precision, recall, f-measure, kappa and different error statistics are observed. As a part of the future work we have planned to increase our sample space with different varieties.

References

Ricardo A Calix, Leili Javadpour, Mehdi Khazaeli, and Gerald M Knapp. 2013. Automatic detection of nominal entities in speech for enriched content search. In *FLAIRS Conference*.

Amit Goyal, Ellen Riloff, and Hal Daumé III. 2010. Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 77–86. Association for Computational Linguistics.

Nuno Mamede and Pedro Chaleira. 2004. Character identification in children stories. In *Advances in natural language processing*, pages 82–90. Springer.

Apurba Paul and Dipankar Das. 2017. Identification of character adjectives from mahabharata. In *RANLP*.

Josep Valls-Vargas, Santiago Ontanón, and Jichen Zhu. 2013. Toward character role assignment for natural language stories. In *Proceedings of the Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 101–104.

Josep Valls-Vargas, Santiago Ontanón, and Jichen Zhu. 2014. Toward automatic character identification in unannotated narrative text. In *Seventh Intelligent Narrative Technologies Workshop*.

Josep Valls-Vargas, Jichen Zhu, and Santiago Ontañón. 2015. Narrative hermeneutic circle: Improving character role identification from natural language text via feedback loops. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Nees Jan Van Eck, Ludo Waltman, Ed CM Noyons, and Reindert K Buter. 2010. Automatic term identification for bibliometric mapping. *Scientometrics*, 82(3):581–596.

Neural Networks for Semantic Textual Similarity

Derek S. Prijatelj
Duquesne University
Pittsburgh, PA 15282
prijateljd@duq.edu

Jonathan Ventura
University of Colorado
Colorado Springs, CO 80918
jventura@uccs.edu

Jugal Kalita
University of Colorado
Colorado Springs, CO 80918
jkalita@uccs.edu

Abstract

Complex neural network architectures are being increasingly used to learn to compute the semantic resemblances among natural language texts. It is necessary to establish a lower bound of performance that must be met in order for new complex architectures to be not only novel, but also worthwhile in terms of implementation. This paper focuses on the specific task of determining semantic textual similarity (STS). We construct a number of models from simple to complex within a framework and report our results. Our findings show that a small number of LSTM stacks with an LSTM stack comparator produces the best results. We use SemEval 2017 STS Competition Dataset for evaluation.

1 Introduction

Scholars have attempted to capture the semantics in natural language texts in a formal manner for centuries. Even today, the true meaning of a word can neither be quantified nor computed, but methods have been developed to express meaning numerically in terms of co-occurrence and association. This is called distributional semantics, and it plays a key role in current approaches to representation of meaning, where although the actual meaning remains unknown, computations can be performed with words or phrases that share the same usage, and therefore, have similar meaning. The theoretical foundation lies in the so-called *distributional hypothesis*, which states that words that share the same context tend to share similar meaning (Harris, 1954). This hypothesis, which is claimed to hold true for words, has been used to obtain vector representations or embeddings for words (Mikolov et al., 2013; Le and Mikolov, 2014). Building on such word embeddings, various methods have been proposed to obtain the meaning of phrases, sentences, paragraphs and whole texts. These include complex linear-algebra based approaches, and more recently a variety of neural network architectures (Le and Mikolov, 2014; Kiros et al., 2015). This research concerns itself specifically with the semantic representation of sentences, and compares the different representations in the task of semantic textual similarity matching.

Semantic textual similarity matching is the task of determining the resemblance of the meanings between two sentences. The dataset used for this task is SemEvals' 2017 Semantic Textual Similarity corpus[1] [2]. The task specifically is to output a continuous value on the scale from [0, 5] that represents the degree of semantic similarity between two given English sentences, where 0 is no similarity and 5 is complete similarity. In terms of machine learning, this is a regression problem. The 2017 STS corpus contains 1186 English sentence pairs with a corresponding rating and 249 pairs as the test set. The test set has been labeled with the average of multiple human expert ratings that SemEval calls the "gold standard". The distribution of ratings is stated to be as uniform throughout as it could be, and the ratios of ratings for the test set are similar to the training set's ratings.

The models that are examined in this re-

[1] http://alt.qcri.org/semeval2017/task1/index.php?id=data-and-tools

[2] http://ixa2.si.ehu.eus/stswiki/index.php/Main_Page

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 456–465

search are simple neural network architectures compared to some of the more complicated models that are popular in recent natural language processing research (Wan et al., 2016; Wu et al., 2017b; Fu et al., 2016; Guo et al., 2016; Liu et al., 2016a; Liu et al., 2017b; Liu et al., 2017a; Liu et al., 2016b). Examining the simple neural network architectures better posits a perspective on creating new architectures for practical applications. If a simple architecture can perform equivalently or better than a more complex model being proposed, the new model is simply a new way to accomplish a task using a resource-hungry method. Our simple models use perceptrons to simple LSTMs and bidirectional LSTMs and are evaluated on the STS task. The major components in these models are the pre-trained word vectors, the sentence embeddings, and the comparator of the two sentence embeddings that performs the regression.

2 Related Work

Pursuing better semantic representation for phrases and sentences, and the use of such representation to solve natural language processing tasks have become popular due to the influence of distributional semantics, in particular representations obtained using artificial neural networks.

2.1 Semantic Representation

Mikolov invigorated the interest in distributional semantics with his team's creation of Word2Vec, a means for representing the co-occurrences of words in written text as elements in a vector space (Mikolov et al., 2013). This method is fairly successful, but by its very nature, it creates embeddings for single words (and common short phrases) only. Stanford University developed another method of computing the distributional semantics of words, and this method is known as GloVe. GloVe is similar to Word2Vec in that it computes the co-occurrence frequency of words and creates a vector of a specified dimension to represent a word, but the methods they use are more linear-algebra based (Pennington et al., 2014). Either may be used for natural language processing tasks depending on preference or performance of the pre-trained word embeddings. In this research, 300 dimensional GloVe word vectors are used as the initial state of the word vectors.

Various methods have been developed to embed a sentence represented by a sequence of words, each with its own vector. Some of these methods involve the use of neural networks, including, but not limited to, LSTMs and their variations (Wan et al., 2016; Palangi et al., 2016; Chen et al., 2016; Liu et al., 2017b). Most of these methods compute sentence representations using methods similar to those applied for word embeddings or as direct extensions of such methods (Kiros et al., 2015; Le and Mikolov, 2014). (Arora et al., 2016) proposed a "simple but tough-to-beat baseline for sentence embeddings" called the SIF embedding method. SIF involves taking the average of all the word vectors in a sentence and removing the first principal component. Arora et. al have reported it to be a satisfactory baseline and it has been found to provide strong results for many tasks.

2.2 Semantic Matching

A recently developed neural net approach to semantic matching is the Matrix Vector-LSTM (MV-LSTM) (Wan et al., 2016). MV-LSTM finds the paired sentence embeddings by processing the respective lists of word vectors individually through separate bidirectional LSTMs. The resulting sentence embeddings are then compared via a Matrix Vector, otherwise known as a similarity tensor. The important features of the similarity tensor are obtained through k-max pooling and the k-max pools are processed via a multilayered perceptron.

There exist other recent architectures for the semantic matching of sentences and they have been used in the recent SemEval STS 2017 competition. These architectures are listed in Table 3. ECNU accomplishes the STS task by combining traditional natural language processing methods with modern deep learning through the use of an ensemble of classifiers to average the two different parts' scores (Tian et al., 2017). This method attempts to use the best of both methods to overcome the limitations in each other. An ensemble of classifiers is a method of using various classifiers, in this case the traditional and

deep learning methods, and combining their votes via some mathematical method, which in ECNU's case is simple averaging. (Henderson et al., 2017) create the MITRE model, which also uses an ensemble, but instead has 5 systems in their ensemble. Their systems include a simple bag-of-words model, a recurrent convolutional neural network, an enhanced BiLSTM inference model, alignment measures of strings, and the open source TakeLab Semantic Text Similarity System (Šarić et al., 2012). Thus, both the MITRE and ECNU systems use ensembles in an attempt to harness the wisdom of the individual components in their final classification for the STS task.

The BIT model makes use of WordNet to create what is called a "semantic information space (SIS)" (Wu et al., 2017a). SIS attempts to obtain unique meaning of words by establishing the shared words of a certain meaning as a unique information space. WordNet (Miller, 1995) is a human created database that defines how English words share meaning, similar to an extended thesaurus. BIT attempts to leverage the knowledge in WordNet entered by actual humans to create a better system and achieve competitive results. HCTI uses a convolutional neural network in order to handle the STS task (Yang, 2017). The convolutional neural network is simple, yet yields competitive results.

FCICU is a model for solving the STS task through the use of a sense-based and surface-based alignment similarity method coupled with an existing semantic network (Hassan et al., 2017). FCICU uses BabelNet (Navigli and Ponzetto, 2012) and an alignment method to perform multilingual tasks in the SemEval 2017 competition. FCICU is an unsupervised model.

The DT_TEAM's model (Maharjan et al., 2017) uses Support Vector Regression (Smola and Schölkopf, 2004), Linear Regression, and Gradient Boosting Regressor with various features that are carefully selected. DT_TEAM's model uses many methods to generate the features on which their model depends. The DT_TEAM's performance in the English-English STS task was second overall with highly competitive scores. The ITNLPAiKE model (Liu et al., 2017c) also uses a Support Vector Regression model with feature engineering. The features ITNLPAiKE uses are ontology based, word embedding based, corpus based, alignment based and literal based features. The ITNLPAiKE model had relatively competitive results in the English-English STS task, and the authors found that the ontology, word embedding, and alignment based features were the most beneficial features that they tested.

3 Examined Models

We examine a number of models that all share the same architecture: Pre-trained word embeddings, a sentence embedding component, and a comparator component. Figure 1 depicts this shared architecture. The sentence embedding component takes the sequence of word vectors that represents a sentence and combines them into a single vector that represents the meaning of the original sentence. The comparator component is the part of the model that evaluates the similarity between the two sentence vectors and performs regression to output the sentence pair's similarity score on a continuous inclusive scale from 0 to 5. For all components and individual neural network units of the model, ELU activations (Clevert et al., 2015) are used. The initial weights of each unit are randomly initialized using the He normal distribution (He et al., 2015). For all models, RMSprop (Tieleman and Hinton, 2012) is used as the optimizer with a learning rate of 1e-4. Mean squared error is the loss function for all models as well. The metrics that are calculated are mean squared error, and the Pearson correlation coefficient (PCC), or Pearson R. The SemEval STS competition uses the PCC as the primary metric of a model's performance.

We examine different sentence embeddings, as well as semantic matching processes as the comparator component of the models. These methods are compared to modified versions of the MV-LSTM. A modified version also replaces the similarity tensor with Euclidean distance to establish an understanding of the simplified models' performance; its results are not reported. One of the models (called the L2-LSTM) uses of bidirectional LSTMs for learning the sentence embeddings from the paired

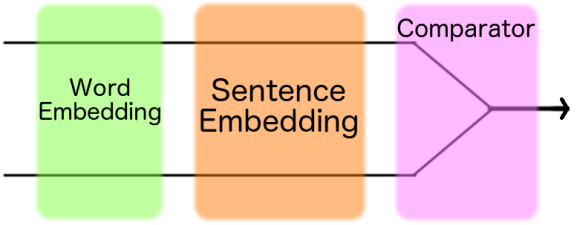


Figure 1: The overall architecture of the simple models for the STS task. Two sentence strings are are the inputs and one float in the range [0, 5] is the output.

list of word vectors. We keep the multi-layered perceptron at the end of the comparator component for most models.

3.1 Pre-Trained Word Vectors

The models start with the input of two sentences represented by strings. The words in the sentences are replaced by word vectors using the provided pre-trained word embeddings, which in this case is a set of GloVe word vectors. This specific set of word vectors have 300 dimensions and were pre-trained on 840 billion tokens taken from Common Crawl[3]. Different pre-trained word vectors may be used in place of this specific pre-trained set. After being embedded into the pre-trained word vectors, the data is randomly shuffled and then sent to the sentence embedding component.

3.2 Sentence Embedding

The model component is responsible for taking a list of word vectors that represent a sentence and embedding them into a single vector to represent the entire sentence. The sentence vector compresses the size of the data that represents the sentence, but synthesizes important semantic information contained in the sentence.

3.2.1 Smooth Inverse Frequency (SIF)

(Arora et al., 2016) propose a method of sentence embedding called smooth inverse frequency (SIF) as a simple baseline for all sentence representations to surpass. The method involves taking the mean of all word vectors in a list and removing the first principal component. They found that this simple method for sentence embedding creates satisfactory results. The hypothesis is that SIF removes the effect of most common words that occur in documents, and is akin to removing stop words from consideration. SIF serves as the method of sentence representation tested in all models in this research.

The formal mathematical representation of SIF is as follows: $v_s = \frac{1}{s}\sum_{w \in s} \frac{a}{a+p(w)} v_w$ where $a = \frac{1-\alpha}{\alpha Z}$, $v_s = v_s - uu^\top v_s$, where s is the current sentence, w represents a word in the sentence, v_w is the vector representation of the word, α is a hyperparameter, Z is the normalizing constant (the partition function) that is roughly the same in all directions, p represents the estimated probabilities of the words, and u is the calculated first principal component.

3.2.2 LSTM

Sentences are sequences of words where order matters and each word may modify any other's meaning despite their location in the sentence. Given that sentences are sequences, it is only natural to use the version of the recurrent neural network known as the LSTM. The version of the LSTM used throughout model is based on the original from (Hochreiter and Schmidhuber, 1997). This sentence embedding component consists of a single LSTM per sentence with a number of hidden units in parallel equal to that of the word embedding's number of dimensions.

The traditional LSTM used is defined as follows:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_c(c_t)
\end{aligned}
$$

where x_t is the input vector, h_t is the output vector, c_t is the cell state vector, W, U and b are the parameter matrices, and vectors f_t, i_t, and o_t are the forget, input, and output gate vectors respectively. σ represents the activation functions where σ_g is an ELU function and σ_c is the hard sigmoid. Figure 2 depicts this LSTM block architecture.

[3] https://nlp.stanford.edu/projects/glove/

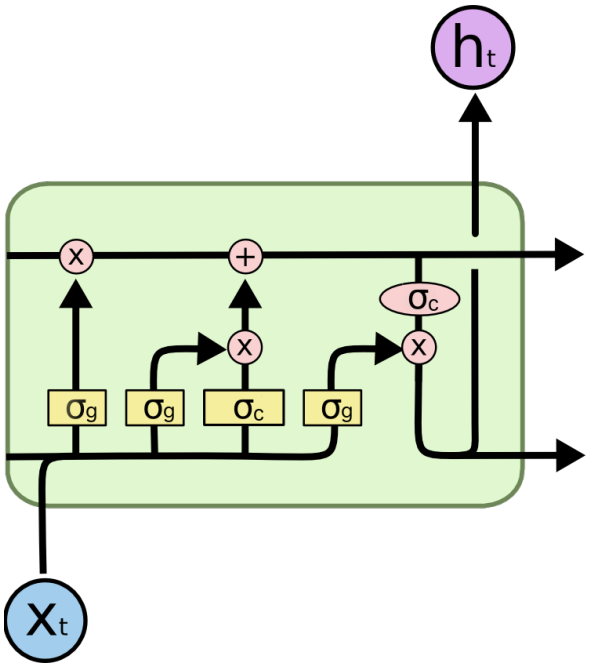


Figure 2: Example of the traditional LSTM architecture with the modified activation functions. The two arrows exiting from the side are the LSTM's recurrent connections. This image was modified from Christopher Olah's original image[5].

3.2.3 Stacked LSTMs

The stacked LSTMs' construction is the same as the the single LSTM, except that instead of one LSTM per sentence there are two stacks of LSTMs of equal length. All hyper-parameters are the same otherwise. Various sized stacks of LSTMs are experimented with, including 2, 3, 4, 5, and 10. Multiple LSTMs should be able to capture the kernels of meaning in a sentence. As stated by (Palangi et al., 2016), the higher the number of LSTMs in the stack, the better the predicted performance of the sentence embedding.

A stack of LSTMs is simply multiple LSTMs whose output is the input to the next LSTM in the stack. The version of LSTM stacks used in this research is sequential, meaning that the processed sequence data from the LSTM is saved and outputted to the next LSTM. The final LSTM of the stack outputs a matrix with the dimensions of batch size, and time steps. Figure 3 indicates the input and output dimensions previously described.

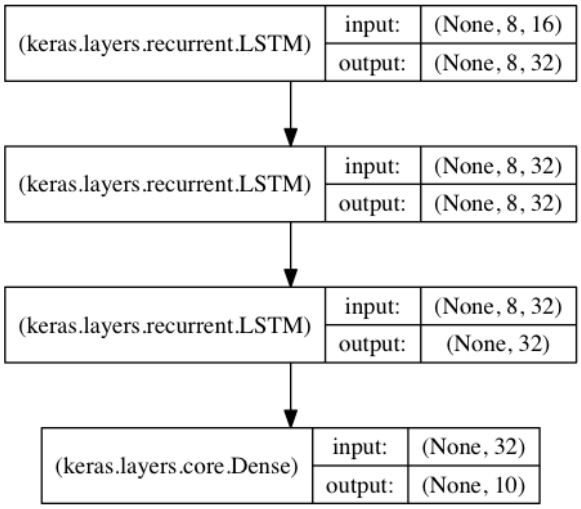


Figure 3: Example of the input and output dimensions of the layers in a small stack of LSTMs as depicted by the Keras API at `https://keras.io/getting-started/sequential-model-guide/`.

3.3 Comparator

The comparator examines the two sentence embeddings and performs regression on them to find a continuous value on the inclusive range from 0 to 5. This continuous value indicates the level of similarity between the two sentences, where 0 is no semantic similarity and 5 is complete semantic similarity.

3.3.1 Perceptron

The simplest of all the comparators, the perceptron with ELU as its activation is used as the regression operation. The weights are initialized at random using the He normal distribution. The outputs from the sentence embeddings are concatenated and sent to a fully connected dense layer, which then connects to a single output node.

3.3.2 LSTM

In order to learn the relationship between the words in the two sentences, an LSTM takes the concatenated sequence output from the two LSTM sentence embedding components. This single LSTM performs the regression on the two embeddings and learns how the two embeddings relate to one another.

3.3.3 Stacked LSTMs

Applying the reasoning behind deep LSTM stacks as proposed by (Palangi et al., 2016),

a stack of LSTMs is used as the comparator of LSTM sentence embeddings. The process is the same as the single LSTM comparator, but instead with a stack of LSTMs. Varying sizes of stacks are used, but match the size of the LSTM stacks in the sentence embedding component.

3.4 Simplified MV-LSTM: L2-LSTM

Unlike the other simple models that come in parts, this model comes together as a whole. A simplified version of the MV-LSTM from (Wan et al., 2016) is also tested among the simple models. This model matches the MV-LSTM exactly except for the similarity tensor which is replaced with an Euclidean distance calculation to compare the similarity to the two sentence embeddings. Bidirectional LSTMs are used for the sentence embeddings and the Euclidean distance is followed by a multilayered perceptron with 3 layers that cuts their density in half from the previous layer. The first layer has 5 nodes. This simplified version of the MV-LSTM is referred to as the L2-LSTM.

4 Implementation

The implementation was done using Keras[6] on an Ubuntu system with GPU support. The training data is given to the neural networks with mean squared error as their loss, due to this being a regression problem. The model is evaluated using cross-validation.

5 Evaluation Process

The STS Benchmark comprises a selection of the English datasets used in the STS tasks organized in the context of SemEval between 2012 and 2017. The selection of datasets includes text from image captions, news headlines and user forums. There are a total of 1186 ranked sentence pairs from various domains such as image captions, Twitter news, questions, answers, headlines, plagiarism, and post-editing. Table 1 shows a few pairs of sentences and their gold standard STS rankings from this dataset.

Each model is evaluated on the sentence similarity task. The results of various models are compared in terms of Pearson Correlation Coefficient, as mentioned earlier. The Pearson Correlation Coefficient is as follows:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where n is the number of samples, x_i and y_i are the single samples indexed with i, $\bar{x}$ and $\bar{y}$ are the two samples' respective means.

6 Results

The results indicate that models with a better capacity for memory storage are better suited for solving the STS task optimally. The simplified MV-LSTMs also perform approximately the same as a perceptron, and thus should be discarded from use in practical application for the STS task. However, these are only simplified versions of the MV-LSTM.

6.1 LSTM

The single LSTM embeddings for both the perceptron comparator and the single LSTM comparator performed worse than any of the models that included a stack of LSTMs. This indicates that the memory of a single LSTM compared to that of a stack of LSTM is unable to learn the semantic essence of a sentence. This encourages the use of models with increased memory due to their ability to learn important semantic features of a sentence.

6.2 Stacked LSTMs

The stacked LSTMs performed the best overall with the paired stack of 10 LSTMs for embedding and a perceptron comparator as the best of all LSTM stack embedding and perceptron comparator models. The stack of 2 LSTMs with a stack of 2 LSTMs as the comparator performed the best out of all of the models with a .05 lead over the second place model, the stack of 10 LSTMs and perceptron model. The success of the LSTM stacks indicates that these models are able to learn kernels of meaning in the sentences and compare them correctly to one another. The quality performance from these models raise the standards for newer, more complex models for the STS task.

6.3 Simplified MV-LSTM: L2-LSTM

The L2-LSTM performed worse than any of the other models, except for the perceptron

[6] `https://keras.io/`

Rank	Sentence 1	Sentence 2
3	In the US, it will depend on the school.	It really depends on the school and the program.
2	I did this one time as well.	I have this habit as well.
5	You do not need to worry.	You don't have to worry.
3	I remained under the banyan tree, exhausted by my daily ritual of dragooning the men every two hours.	I remained under the banyan tree, exhausted by my daily ritual of herding the cats every two hours.
0	You need to read a lot to know what you like and what you don't.	You should tell a good story and sometimes you have to tweak reality/history to do so.
1	If you are not sure how to do it, don't do it at all.	If not, don't do that and spend that time with something you like to do.

Table 1: Example sentence pairs from the SemEval STS 2017 dataset

Simple Models' Mean Pearson R across 10 K-Fold Cross Validation		
Model Name	Pearson R	In-Sample Pearson R
2 LSTM Stack and 2 LSTM Stack	**0.8608**	0.9963
10 LSTM Stack and Perceptron	0.7824	0.9082
2 LSTM Stack and Perceptron	0.7595	0.8757
3 LSTM Stack and Perceptron	0.7235	0.8275
4 LSTM Stack and Perceptron	0.7150	0.8269
5 LSTM Stack and Perceptron	0.4538	0.6465
1 LSTM and Perceptron	0.4301	0.5445
1 LSTM and 1 LSTM	0.4163	0.9902
L2-LSTM 50 epochs	0.2740	0.3537
SIF and Perceptron	0.2214	0.8795
L2-LSTM 100 epochs	0.2183	0.3066

Table 2: The mean Pearson R out-of-sample and in-sample from k-fold cross validation where k = 10. The LSTM and LSTM Stack embeddings were all computed with 50 epochs. The SIF embedding and perceptron comparator were calculated with 100 epochs. The model names are ordered by embedding component and comparator, except for the L2-LSTM model which is combined embedding and comparator. In-Sample Pearson R is the Pearson R of the model evaluated on the data used to train the model.

when compared to the MV-LSTM with 50 epochs. This indicates that either the bidirectional LSTMs are not suitable for learning the semantics between the two sentences, or the similarity comparison with the Euclidean distance is not as effective as the power of the learning the sequences with LSTMs. Given its performance roughly matches that of a perceptron, the L2-LSTM is a model not to be used given its similar performance to, but greater complexity than the perceptron.

6.4 Comparison with SemEval STS 2017 Results

We compare our results with papers and systems from SemEval STS 2017 (Cer et al., 2017). These papers and systems use Pearson Correlation Coefficient also for evaluation. Table 3 presents the results of the top-performing systems from the SemEval 2017 contest.

We must note that the papers published in SemEval 2017 used the Training, Development, Evaluation datasets whereas we performed 10-fold cross-validation. Thus, the results are not quite comparable, but we feel that cross-validation may be a better way to

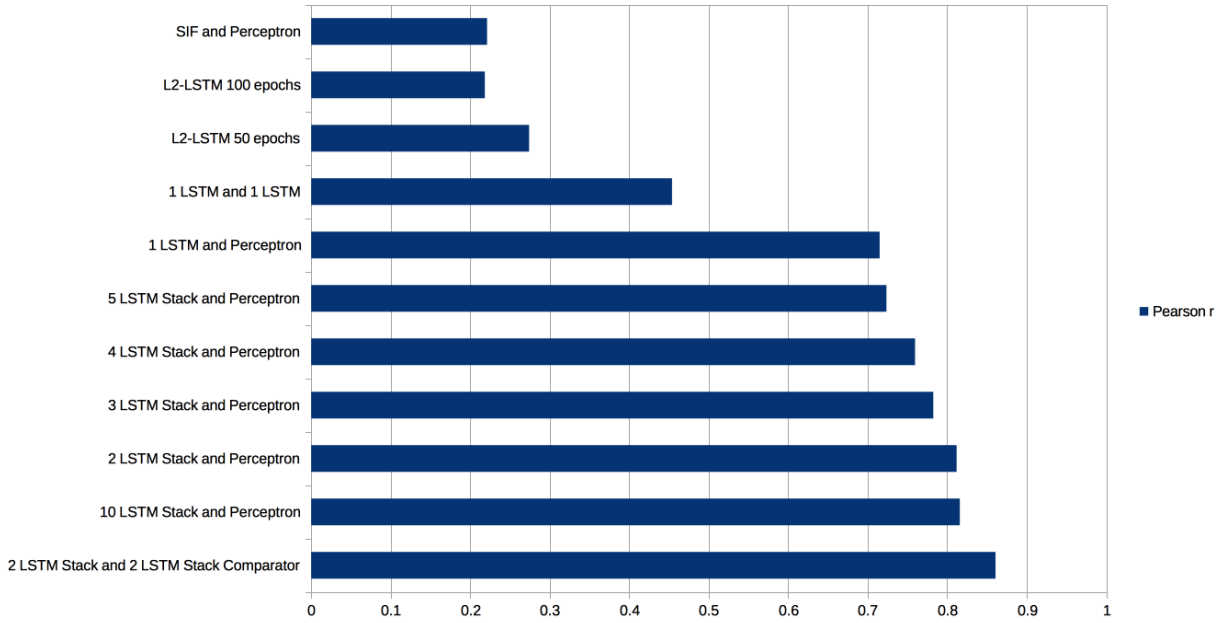


Figure 4: The mean Pearson R across all test and validation sets in k-fold cross validation where k = 10.

Team	Paper	PCC Results (3 or fewer runs)
ECNU	(Tian et al., 2017)	.8515, .8181, .8387
BIT	(Wu et al., 2017a)	.8400, .8161, .8222
HCTI	(Yang, 2017)	.8113, .8156
MITRE	(Henderson et al., 2017)	.8053, .8048
FCICU	(Hassan et al., 2017)	.8272, .8280, .8217
RTV	none	.8541, .8541, **.8547**
DT_TEAM	(Maharjan et al., 2017)	.8536, .8360, .8329
ITNLPAiKE	(Liu et al., 2017c)	.8231, .8231, .8159

Table 3: Results of SemEval STS 2017 Competition in terms of Pearson Correlation Coefficient, as published in (Cer et al., 2017)

evaluate than using a hand-selected Evaluation dataset, but this point remains debatable. In spite of this discrepancy, we claim that one of our methods, namely "2 LSTM Stack and 2 LSTM Stack" shows better performance than the approaches in Table 3.

7 Further Research

The performance of the simplified MV-LSTMs bring into question the adequacy of the original MV-LSTM for the STS task. The next step is to evaluate the performance of the MV-LSTM in the STS task and compare it to that of the LSTM stacks. The results indicate that models with a higher capacity for memory are better suited to learn the semantic representation of the sentences and appropriately compare them. These results encourage further research in memory augmented neural networks for use in learning the semantics of natural languages. Exploring the implementation of more complicated memory augmented neural networks, such as the DNC model created by (Graves et al., 2016), is the next step in pursuing better performance in sentence embedding and semantic textual similarity matching.

8 Conclusion

The performances of various simple neural network models have been examined on the task of semantic textual similarity matching using SemEval's provided dataset. The model to perform the best with a Pearson correlation of 0.8608, based on the mean k-fold cross

validation, is the model where a stack of 2 LSTMs embedded the sentences and were then compared with another stack of 2 LSTMs after concatenating the two sentence embedding stacks' sequences output. This supports the findings that natural language tasks are sequence problems where the elements in the sequence have interconnected relatedness, in which neural networks with memory are better at learning. Our findings also suggest that there exist coherent subgroups of words in a sentence whose meanings can learned and composed to obtain the unique meaning of a sentence. This supports the findings that MV-LSTM also obtains. The evaluation of these simple models for semantic textual similarity serves as the lower bound to compare all other models that have increased complexity in their design. All future researchers should ensure that their new model architectures surpass these lower bounds.

References

[Arora et al.2016] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings. *International Conference on Learning Representations*.

[Cer et al.2017] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

[Chen et al.2016] Jifan Chen, Kan Chen, Xipeng Qiu, Qi Zhang, Xuanjing Huang, and Zheng Zhang. 2016. Learning word embeddings from intrinsic and extrinsic views. *arXiv preprint arXiv:1608.05852*.

[Clevert et al.2015] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

[Fu et al.2016] Jian Fu, Xipeng Qiu, and Xuanjing Huang. 2016. Convolutional deep neural networks for document-based question answering. In *International Conference on Computer Processing of Oriental Languages*, pages 790–797. Springer.

[Graves et al.2016] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

[Guo et al.2016] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. Semantic matching by non-linear word transportation for information retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 701–710. ACM.

[Harris1954] Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

[Hassan et al.2017] Basma Hassan, Samir Abdel-Rahman, Reem Bahgat, and Ibrahim Farag. 2017. Fcicu at semeval-2017 task 1: Sense-based language independent semantic textual similarity approach. *Proceedings of SemEval-2017*.

[He et al.2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852.

[Henderson et al.2017] John Henderson, Elizabeth Merkhofer, Laura Strickhart, and Guido Zarrella. 2017. Mitre at semeval-2017 task 1: Simple semantic similarity. *Proceedings of SemEval-2017*.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Kiros et al.2015] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

[Le and Mikolov2014] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

[Liu et al.2016a] Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016a. Deep fusion lstms for text semantic matching. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

[Liu et al.2016b] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016b. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

[Liu et al.2017a] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017a. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*.

[Liu et al.2017b] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017b. Dynamic compositional neural networks over tree structure. *arXiv preprint arXiv:1705.04153*.

[Liu et al.2017c] Wenjie Liu, Chengjie Sun, Lei Lin, and Bingquan Liu. 2017c. Itnlp-aikf at semeval-2017 task 1: Rich features based svr for semantic textual similarity computing. *Proceedings of SemEval-2017*.

[Maharjan et al.2017] Nabin Maharjan, Rajendra Banjade, Dipesh Gautam, Lasang J Tamang, and Vasile Rus. 2017. Dt team at semeval-2017 task 1: Semantic similarity using alignments, sentence-level embeddings and gaussian mixture model output. *Proceedings of SemEval-2017*.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Miller1995] George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

[Navigli and Ponzetto2012] Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

[Palangi et al.2016] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[Šarić et al.2012] Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. Takelab: Systems for measuring semantic text similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 441–448. Association for Computational Linguistics.

[Smola and Schölkopf2004] Alex J Smola and Bernhard Schölkopf. 2004. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

[Tian et al.2017] Junfeng Tian, Zhiheng Zhou, Man Lan, and Yuanbin Wu. 2017. ECNU at SemEval-2017 Task 1: Leverage kernel-based traditional NLP features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity.

[Tieleman and Hinton2012] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

[Wan et al.2016] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[Wu et al.2017a] Hao Wu, Heyan Huang, Ping Jian, Yuhang Guo, and Chao Su. 2017a. Bit at semeval-2017 task 1: Using semantic information space to evaluate semantic textual similarity. *Proceedings of SemEval-2017*.

[Wu et al.2017b] Zongda Wu, Hui Zhu, Guiling Li, Zongmin Cui, Hui Huang, Jun Li, Enhong Chen, and Guandong Xu. 2017b. An efficient wikipedia semantic matching approach to text document classification. *Information Sciences*, 393:15–28.

[Yang2017] Shao Yang. 2017. HCTI at semeval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity.

Open Set Text Classification using Convolutional Neural Networks

Sridhama Prakhya[†], Vinodini Venkataram[‡] and Jugal Kalita[‡]

[†]School of Engineering & Technology, BML Munjal University, Gurugram, India
[‡]Department of Computer Science, University of Colorado Colorado Springs, USA
[†]sridhama@sridhama.com
[‡]{vvenkata,jkalita}@uccs.edu

Abstract

In a *closed world* setting, classifiers are trained on examples from a number of classes and tested with unseen examples belonging to the same set of classes. However, in most real-world scenarios, a trained classifier is likely to come across novel examples that do not belong to any of the known classes. Such examples should ideally be categorized as belonging to an unknown class. The goal of an *open set* classifier is to anticipate and be ready to handle test examples of classes unseen during training. The classifier should be able to declare that a test example belongs to a class it does not know, and possibly, incorporate it into its knowledge as an example of a new class it has encountered. There is some published research in open world image classification, but open set text classification remains mostly unexplored. In this paper, we investigate the suitability of *Convolutional Neural Networks* (CNNs) for open set text classification. We find that CNNs are good feature extractors and hence perform better than existing state-of-the-art open set classifiers in smaller domains, although their open set classification abilities in general still need to be investigated.

1 Introduction

With increasing amounts of textual data being generated by various online sources like social networks, text classifiers are essential for the analysis and organization of data. Text classification usually consists of training a classifier on a labeled text corpus where individual examples belong to one or more classes based on their content, and then using the trained classifier to place unseen examples in one of these classes. Popular text classification applications include spam filtering, sentiment analysis, movie genre classification, and document classification. Traditional text classifiers assume a *closed world* approach. In other words, the classifier is implicitly expected to be tested with examples from the same classes with which it was initially trained. However, such classifiers fail to identify and adapt when examples of previously unseen classes are presented during testing. In real-world scenarios, a robust trained classifier should be able to recognize examples of unknown classes and accordingly update its learned model. This is known as the *open world* approach to classification. Most research in open set classification has been in the computer vision domain, primarily in handwriting recognition (Jain et al., 2014), face recognition (Li and Wechsler, 2005; Scheirer et al., 2013), object classification (Bendale and Boult, 2015; Bendale and Boult, 2016) and computer forensics (Rattani et al., 2015). Open set classification is important in computer vision since the number of classes to which a seen object can belong to is almost limitless and datasets are available with training samples belonging to thousands of classes. Nevertheless, open set classification is important in natural language processing as well. An example of an open world text classification scenario is authorship attribution, where each author happens to be a class. An open set text classifier must recognize the author of a document to be one of the known ones when appropriate. Importantly, the classifier should also explicitly recognize when it fails to classify an unseen document as written by one of the known authors. Whether it is for historical or fictional works from the past, or emails, social media posts or leaked political documents, open set classification may be immensely helpful.

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 466–475

In the recent past, many-layered Artificial Neural Networks (ANN) or deep learning techniques (Goodfellow et al., 2016) have become popular in Computer Vision and Natural Language Processing. This is mainly attributed to the increase in performance compared to standard machine learning techniques. As discussed later, current open set text classifiers do not rely on deep learning models. They employ either a clustering-based approach (Doan and Kalita, 2017) or a modified Support Vector Machine (SVM) (Fei and Liu, 2016). To this end, we explore the possibility of using a CNN for open set text classification and compare it to existing techniques.

2 Related Work

To allow for the possibility that the set of classes is open or expandable during deployment, the classification algorithms need to be adaptive. (Scheirer et al., 2013) combine empirical risk and open space risk due to the existence of a space in which classification probabilities are not currently known. Empirical risk comes from actual examples being misclassified by a trained classifier, and the open space risk recognizes the fact that the presence of unknown classes is likely to introduce errors into classification decisions. Their model reduces the risk by introducing parallel hyperplanes, one near the class boundary and another far from it to introduce slabs of subspaces for the classes, and then develops a greedy optimization algorithm that modifies a linear SVM and moves the planes incrementally. This work was extended to multi-class open set classification by introducing what (Scheirer et al., 2014) call a Compact Abating Probability (CAP) model. They build a classifier called W-SVM using properties of Extreme Value Theory for calibration of scores produced by 1-class and binary SVMs. Extreme Value Theory (EVT) (Smith, 1990; De Haan and Ferreira, 2007; Castillo, 2012) is usually used to deal with and predict rare events or values that occur at the tails of distributions. The unnormalized probability of inclusion for each class is estimated by fitting a Weibull distribution (Sharif and Islam, 1980) over the positive class scores from SVM classifiers. The assumption here is when a trained classifier cannot classify an example as belonging to any of the known classes, it is a case of "failure" of the classifier and is deemed unusual. (Jain et al., 2014) also use EVT to formulate the open set classification problem as one of modeling positive training data at the decision boundary. They introduce a new algorithm called the P_i-SVM for estimating the unnormalized posterior probability of class inclusion. Their approach is different from the one introduced by (Platt and others, 1999) of taking SVM outputs and converting them to probabilities by fitting a sigmoid function to the SVM scores.

(Bendale and Boult, 2015) present an approach to minimize the weighted sum of empirical risk and open set risk using thresholding sums of monotonically decreasing recognition functions, and use their approach to extend the Nearest Centroid Classifier (NCM) (Rocchio, 1971). This classifier represents classes by the mean feature vector of its elements. An unseen example is assigned a class with the closest mean. The Nearest Non-Outlier (NNO) algorithm (Bendale and Boult, 2015) adapts NCM for open set classification, taking into account open space risk and metric learning. The nearest class mean metric learning (NCMML) (Mensink et al., 2013) approach extends the NCM technique by replacing the Euclidean distance with a learned low-rank Mahalanobis distance. This gives better results than the former as the algorithm is able to learn features inherent in the training data.

All the work mentioned so far have been in the context of computer vision. Work in open set classification for textual data is limited. (Fei and Liu, 2016) use CBS learning (Fei and Liu, 2015) where a document is represented as a vector of similarities from centers of spheres that correspond to individual classes. Around the sphere that represents positive examples of a class, they draw a slightly bigger sphere to provide additional space for a class to accommodate unseen examples. They also use SVM hyperplanes to bound the bigger spheres. The unbounded regions correspond to unknown classes.

The Nearest Centroid Class (NCC) algorithm (Doan and Kalita, 2017) builds upon the NCM, but uses a density-based method following the approach of the clustering algorithm called DBSCAN (Ester et al., 1996). They represent a class not by a sphere but a set of density-connected regions and also consider the centroids of these regions and not the means.

In the context of deep learning, (Bendale and Boult, 2016) adapt a CNN (Krizhevsky et al.,

2012) to perform open set classification in the vision domain. In closed set classification, the final softmax layer of the CNN essentially chooses the output class with the highest probability with respect to all other output labels. Bendale and Boult propose OpenMax, which is a new model layer that estimates the probability of an input belonging to an unknown class instead of softmax. (Ge et al., 2017) adapt OpenMax to generative adversarial networks (GANs) for open set vision problems. There have been no such attempts in the text processing domain.

3 Method

Along the lines of existing open set techniques, our work was also motivated by the Rocchio method (Rocchio, 1971). We wanted to use pretrained word vectors (Mikolov et al., 2013) for open set determination. This led us to perform experiments to see whether simple cosine computation can be used for open set classification. We used a naive approach to construct document vectors by averaging all word vectors (Le and Mikolov, 2014) in a document. We calculated the cosine similarities between the mean of all document vectors and a test example. Due to the similarities being too close (sometimes overlapping), we concluded that calculating cosine similarity at the document level was not suitable for open set classification.

Prior open set text classification models (CBS learning and NCC) do not use artificial neural networks. We decided to pursue a novel approach to open set text classification that relied on a deep learning model, viz. CNNs due to their ability of extracting useful features. Since (Bendale and Boult, 2016) explored the use of CNNs in open set image classification, we started with their approach as the basis and extend the work as necessary. The work of (Kim, 2014) in CNNs for sentence classification helped us arrive at an efficient neural network architecture. Thus, we perform experiments with a single-layer CNN, using the Weibull-modified final layer instead of softmax. We also examine if increasing the number of CNN layers changes performance of open set text classification. We develop a novel ensemble approach to deal with the activations of the penultimate layer of the CNN. The penultimate layer is the focus because this is the layer that contains the real activations for nodes corresponding to the various classes for the problem at hand. Since these are raw activations, in a standard CNN, they are converted into probability-like values by performing the softmax operation.

$$softmax\left(x\right)_i \;=\; \frac{e^{x_i}}{\sum_j e^{x_j}} \qquad (1)$$

However, in our case, there is an unknown class to be considered as well and we do not know the activations or probabilities associated with such an unknown class. Therefore, this softmax layer needs to be modified. (Bendale and Boult, 2016) replace the layer that computes softmax with the so-called OpenMax layer, which uses a learned distance metric taking into account the open set risk.

Our new model uses an ensemble approach to make a decision with the activations in the penultimate layer. Our model is also incremental in nature. This means, the model does not have to be retrained after the introduction of a new unknown class. This is because open set determination happens after training, rather than during or before.

In our experiments discussed here, we compare the performance of our ensemble-based open set text classifier with other open set classifiers that have been previously used for image classification and the methods of (Fei and Liu, 2016) and (Doan and Kalita, 2017), which were used for open set text classification.

3.1 Datasets

For efficacious open world evaluation, we must choose a dataset with a large number of classes. This allows us to hide classes during training. These hidden classes can later be used during testing to gauge the open world accuracy. We use the following two freely available datasets.

- **20 Newsgroups** (McCallum et al., 1998; Slonim and Tishby, 2000) - Consists of 18,828 documents partitioned (nearly) evenly across 20 mutually exclusive classes.
- **Amazon Product Reviews** (Jindal and Liu, 2008) - Consists of 50 classes of products or domains, each with 1,000 review documents.

3.2 Evaluation Procedure

Traditional evaluation (closed set) occurs when the classifier is assessed with data similar to what was learned during training. The number of classes presented during testing is equal to the number

the model was trained on. In open set evaluation, the classifier has incomplete knowledge during the training phase. Examples of unknown classes can be submitted to the classifier during the testing phase. During the training phase, we train the classifiers on a limited number of classes. While testing, we then present the model with additional classes that were not learned during training. We evaluate the performance of the classifier based on how well it identifies these new classes. *"Openness"*, proposed by (Scheirer et al., 2013; Scheirer et al., 2014), is a measure to estimate the open world range of a classifier. This measure is only concerned with the number of classes rather than the open space itself.

$$openness = 1 - \sqrt{(2 \times C_T)/(C_R + C_E)} \quad (2)$$

where:

C_T = number of classes used for training
C_R = number of classes to be recognized
C_E = number of classes used during evaluation/testing

As a special case, when $C_T = C_R = C_E$, the value of *openness* is 0, i.e., it is the case of traditional classification when the numbers of classes trained on, tested on, and recognized are the same.

Accuracy, precision, recall, and F-score are used to measure the closed set performance of our model. These metrics are expanded to the open set scenario by grouping all unknown classes into the same subset. A True Positive is when an example of a known class is correctly classified and a True Negative is when an example of an unknown class is correctly predicted as unknown. False Positives (an unknown class predicted as known) and False Negatives (a known class predicted as unknown) are the two types of incorrect class assignment. Figure 1 shows how *openness* varies with the number of training classes when there are 10 testing classes.

4 Experiments

For all experiments, the CNN-static architecture proposed by (Kim, 2014) is used. We use pre-trained `word2vec`[1] (Mikolov et al., 2013) vectors as our word embeddings. These embeddings are kept static while other parameters of the model are learned. According to the experiments of (Zhang and Wallace, 2015), imposing an L^2 norm constraint on the weight vectors generally does not improve performance drastically. Figures 3 and 4 show the accuracies achieved on the 20 Newsgroups dataset while varying the L^2 norm constraint. Increasing the L^2 norm constraint proved detrimental to the model accuracy. The configuration details of the CNN used in all our experiments are shown in Table 1. Figure 2 shows a depiction of the CNN architecture we implemented. In our case, we use a single static channel instead of multiple channels.

[1] `https://code.google.com/p/word2vec/`

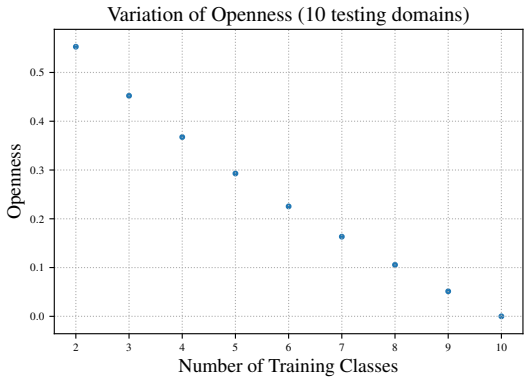


Figure 1: Variation of openness with number of training classes

Table 1: CNN baseline configuration

Description	Values
word embedding	`word2vec`
filter sizes	(3,4,5)
feature maps	100
activation function	ReLU
pooling	1-max pooling
dropout rate	0.5
L^2 norm constraint	0.0

4.1 Multi-layer CNN

In addition to Kim's architecture, we have also experimented with multi-layer CNNs. We used 2 convolutional layers, the initial layer used a kernel of size 3×1, while the second layer used a kernel of size 3×300. The first layer convolves the *same* feature across multiple words of the document. The second layer convolves *all* features (obtained from the previous convolution) across multiple (3 in our case) rows. The motive behind this approach was to extract activation vec-

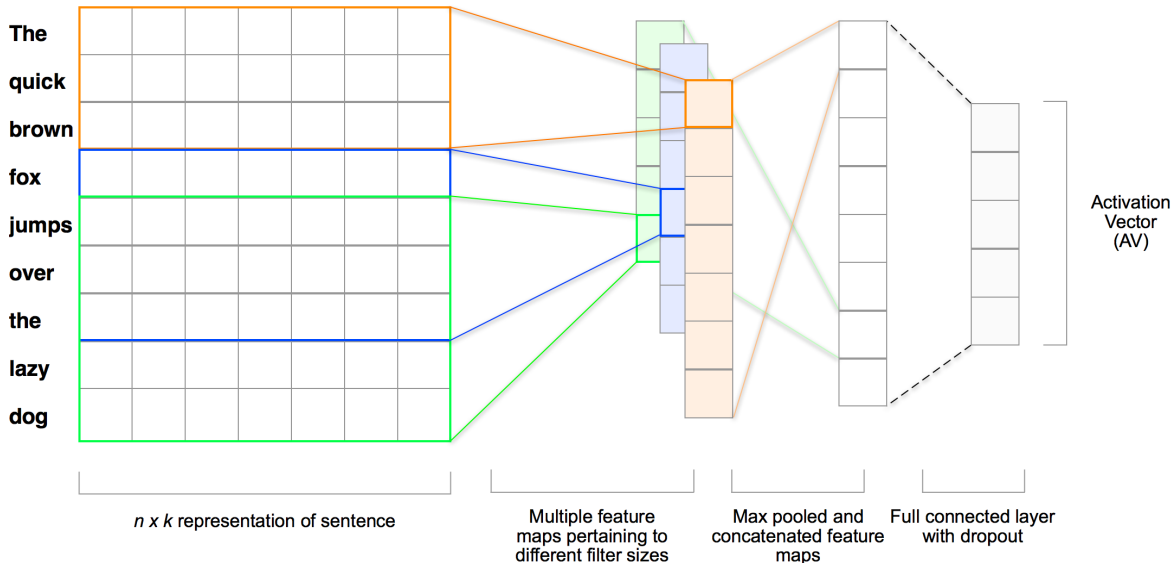


Figure 2: Model architecture with multiple filter sizes (3, 4, 5) for an example sentence

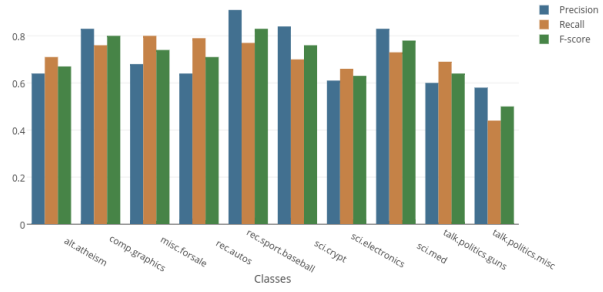


Figure 3: L^2 constraint = 0.0, Model Accuracy: 0.710

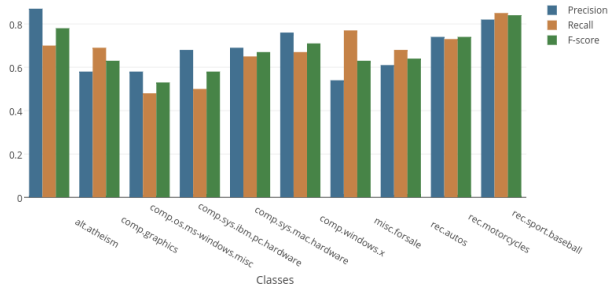


Figure 4: L^2 constraint = 3.0, Model Accuracy: 0.672

tors from the antepenultimate layer, which may represent the document more accurately. Unfortunately, the closed set (trained on 3 classes) accuracy of the muli-layer CNN was around 75%. The accuracy decreased significantly as we increased the number of training classes. A high closed set accuracy is necessary to achieve respectable open set results. Intuitively, the model must have a comprehensive understanding of what it knows. Only then can it be competent enough to classify unknown inputs correctly.

4.2 Ensemble Approach

In our open set classifier, we use an ensemble of approaches to determine whether a test example is from a known class or not. This ensemble includes probabilistic and high dimensional outlier detectors.

4.2.1 Isolation Forest

The isolation forest algorithm (Liu et al., 2008) detects outliers using combinations of a set of isolation trees. Isolation trees recursively partition the data at random partition points with randomly chosen features. Doing so isolates instances into nodes containing only one instance. The heights of branches containing outliers are comparatively less than other data points. The height of the branch is used as the outlier score. The scores obtained from the isolation forest are min-max normalized and calculated for every training class. Examples with scores below a predefined threshold are labelled as unknown. In case of multiple scores above the threshold, the example is assigned to the class with the highest score.

4.2.2 Probabilistic Approach

OpenMax (Bendale and Boult, 2016) is a new model layer based on the concept of Meta-Recognition (Scheirer et al., 2011). For all positive examples of every trained class, we collect the scores in the penultimate layer of our neural

network. We call these scores activation vectors (AV). We deviate from the original OpenMax by finding the k-nearest examples to the centroid of every training class. We refer to these examples as k-Class Activation Vectors (k-CAV). For every example in a training class, we calculate the distances between the respective AV and the k-CAVs. Doing so, results in k distances per AV. We then take the average of these k calculated distances. As the number of classes in our dataset is far less than those used in image classification, the k-CAVs of a class are used represent a class more accurately than a single mean activation vector. This also mitigates the effect of outlier AVs in a class. We observed that when k is around 10, the trade-off between performance and computation time is optimized. Therefore, for all experiments, we fix the value of k = 10.

In our outlier ensemble, we use two distance metrics – Mahalanobis distance and Euclidean-cosine (Eucos) distance (Bendale and Boult, 2016). Ideally, we want a distance metric that can tell us how much an example deviates from the class mean. The Mahalanobis distance precisely does this by giving us a multi-dimensional generalization of the number of standard deviations a point is from the distribution's mean. The closer an example is to the distribution mean, the lower is the Mahalanobis distance. The Mahalanobis distance between point x and point y is given by:

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T C^{-1} (\vec{x} - \vec{y})} \qquad (3)$$

where C is the covariance matrix, among the feature variables calculated a priori. The Euclidean-cosine distance is a weighted combination of Euclidean and cosine distances.

The distances obtained are used to generate a Weibull model for every training class. We use the libMR[2] (Scheirer et al., 2011) *FitHigh* method to fit these distances to a Weibull model that returns a probability of inclusion of the respective class. Figure 5 shows the probabilities of inclusion obtained from the generated Weibull model for 2 training classes belonging to the 20 Newsgroups dataset. As an example deviates more from the class center (k-CAVs), the probability of inclusion decreases.

The sum of all inclusion probabilities is taken as the total closed set probability. Open set probability (OSP) is computed by subtracting the total closed set probability from 1.

$$OSP = 1 - total\ closed\ set\ probability \qquad (4)$$

We then compare the maximum closed set probability and total open set probability. If the total open set probability is greater than the former, we label the example as unknown, otherwise, the example is assigned the class with the highest closed set probability. Parameters like threshold and distribution tail-size can be be adjusted to decrease the open-space risk.

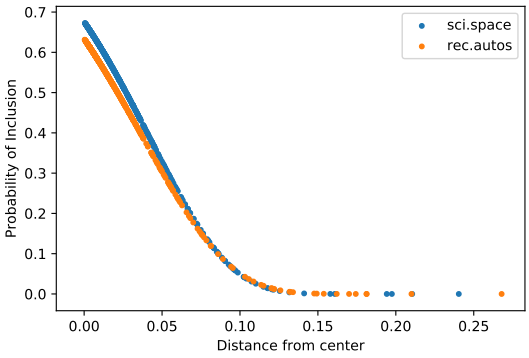


Figure 5: Weibull distribution generated using libMR for two classes belonging to the 20 Newsgroups dataset

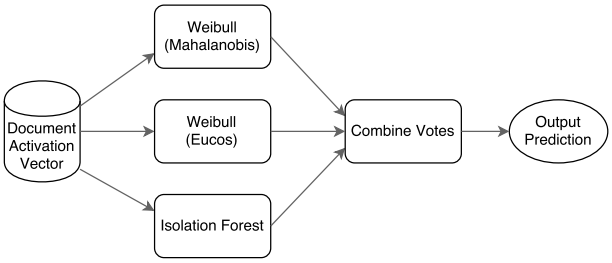


Figure 6: Our ensemble model

We use a voting scheme to combine the three approaches (Mahalanobis Weibull, Eucos Weibull and Isolation Forest), see Figure 6. It has been observed that Mahalanobis and Eucos perform nearly the same. Predictions from the Isolation Forest are usually used as a tie-breaker in case of differing predictions. When all 3 predictions differ, we give the Eucos Weibull the highest priority.

5 Results and Discussion

Open set performance largely depends on the "unknown" classes used during evaluation. This is especially true when classes are not completely exclusive. The activation vectors of similar classes

[2] https://github.com/Vastlab/libMR

Table 2: Experiments on Amazon Product Reviews dataset (10, 20 domains)

Amazon Product Reviews	**10 Domains**			
	25%	**50%**	**75%**	**100%**
our model	**0.797**	**0.753**	0.727	0.821
NCC §	0.61	0.714	**0.781**	0.854
cbsSVM*	0.450	0.715	0.775	**0.873**
1-vs-rest-SVM*	0.219	0.658	0.715	0.817
ExploratoryEM*	0.386	0.647	0.704	0.854
1-vs-set-linear*	0.592	0.698	0.700	0.697
wsvm-linear*	0.603	0.694	0.698	0.702
wsvm-rbf*	0.246	0.587	0.701	0.792
P_i-osvm-linear*	0.207	0.590	0.662	0.731
P_i-osvm-rbf*	0.061	0.142	0.137	0.148
P_i-svm-linear*	0.600	0.695	0.701	0.705
P_i-svm-rbf*	0.245	0.590	0.718	0.774

Amazon Product Reviews	**20 Domains**			
	25%	**50%**	**75%**	**100%**
our model	**0.648**	0.603	0.663	**0.793**
NCC §	0.606	0.657	**0.702**	0.78
cbsSVM*	0.566	**0.695**	0.695	0.760
1-vs-rest-SVM*	0.466	0.610	0.616	0.688
ExploratoryEM*	0.571	0.561	0.573	0.691
1-vs-set-linear*	0.506	0.560	0.589	0.620
wsvm-linear*	0.553	0.618	0.625	0.641
wsvm-rbf*	0.397	0.502	0.574	0.701
P_i-osvm-linear*	0.453	0.531	0.589	0.629
P_i-osvm-rbf*	0.143	0.079	0.058	0.050
P_i-svm-linear*	0.547	0.620	0.628	0.644
P_i-svm-rbf*	0.396	0.546	0.675	0.714

usually overlap in vector space. Similar to (Fei and Liu, 2016; Doan and Kalita, 2017), we conduct our experiments by introducing "unseen" classes during testing. In reality, as the train-test partition can be random, we arbitrarily specify the number of testing domains. For every domain, we report our results using 5 random train-test partitions for each dataset. Both datasets are evaluated on the same number of test classes (10, 20). We also evaluate our model on smaller domains, shown in Table 4. The number of testing classes used during training is varied in quarter-step increments (25%, 50%, 75% and 100%). We take the floor value in case of fractional percentages. Using 100% of the testing classes during training corresponds to closed set classification.

Results of the Amazon Product Reviews and 20 Newsgroups datasets are shown in Tables 2 and 3 respectively. We report only the F-scores due to space constraints. Classifiers used as baselines for comparison are described below.

- **1-vs-rest-SVM** - Standard 1-vs-rest multi-class SVM with Platt Probability Estimation (Platt and others, 1999)
- **1-vs-set-linear** - 1-vs-set machine model proposed by (Scheirer et al., 2013)
- **W-SVM** - Weibull-calibrated SVM (Scheirer et al., 2014)
- **P_i-SVM** - SVM model that estimates the unnormalized posterior probability of class inclusion (Jain et al., 2014)
- **ExploratoryEM** - "Exploratory" version of Expectation-Maximization algorithm (EM) (Dalvi et al., 2013)
- **cbsSVM** - Center-Based Similarity Space SVM (Fei and Liu, 2016)

Table 3: Experiments on 20 Newsgroups dataset (10, 20 domains)

20 Newsgroups	**10 Domains**			
	25%	**50%**	**75%**	**100%**
our model	**0.719**	0.747	0.738	0.864
NCC §	0.652	**0.781**	**0.818**	**0.878**
cbsSVM*	0.417	0.769	0.796	0.855
1-vs-rest-SVM*	0.246	0.722	0.784	0.828
ExploratoryEM*	0.648	0.706	0.733	0.852
1-vs-set-linear*	0.678	0.671	0.659	0.567
wsvm-linear*	0.666	0.666	0.665	0.679
wsvm-rbf*	0.320	0.523	0.675	0.766
P_i-osvm-linear*	0.300	0.571	0.668	0.770
P_i-osvm-rbf*	0.059	0.074	0.032	0.026
P_i-svm-linear*	0.666	0.667	0.667	0.680
P_i-svm-rbf*	0.320	0.540	0.705	0.749

20 Newsgroups	**20 Domains**			
	25%	**50%**	**75%**	**100%**
our model	**0.668**	0.686	0.685	0.787
NCC §	0.635	**0.723**	**0.735**	**0.884**
cbsSVM*	0.593	0.701	0.720	0.852
1-vs-rest-SVM*	0.552	0.683	0.682	0.807
ExploratoryEM*	0.555	0.633	0.713	0.864
1-vs-set-linear*	0.497	0.557	0.550	0.577
wsvm-linear*	0.563	0.597	0.602	0.677
wsvm-rbf*	0.365	0.469	0.607	0.773
P_i-osvm-linear*	0.438	0.534	0.640	0.757
P_i-osvm-rbf*	0.143	0.029	0.022	0.009
P_i-svm-linear*	0.563	0.599	0.603	0.678
P_i-svm-rbf*	0.370	0.494	0.680	0.767

- **NCC** - Nearest Centroid Class model (Doan and Kalita, 2017)

F-score performances of 1-vs-rest-SVM, 1-vs-set SVM, W-SVM, P_i-SVM, and cbsSVM are from study (Fei and Liu, 2016), marked as *. Results pertaining to the Nearest Centroid Class model (NCC) are from study (Doan and Kalita, 2017), marked as §. Our model performs better than cbsSVM and NCC classifiers in smaller domains. Figure 7 shows the activation vectors obtained from models trained on 2 classes plotted in 2-dimensional space. The plots show distinct clusters of activation vectors. We believe the CNN approach effectively isolates documents in smaller domains compared to other SVM-based approaches.

Unlike cbsSVM, our model is an incremental model i.e., we do not have to retrain the model when new unknown classes are introduced. Such models are more viable in real world scenarios.

Table 4: Open set results of Amazon Product Reviews Dataset in smaller domains (3, 4, 5)

Classes Trained on	**Classes Tested on**		
	3	**4**	**5**
2	0.802	0.824	0.808
3	-	0.725	0.763
4	-	-	0.797

6 Conclusion

Our incremental open set approach handles text documents of unseen classes in smaller domains more consistently than existing text classification models, namely CBS learning and the NCC model. This research can prove beneficial when

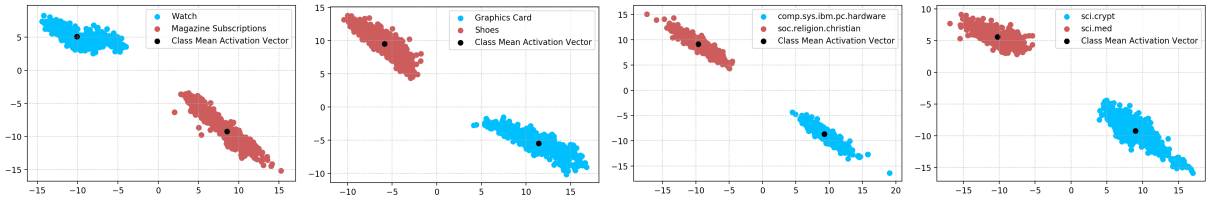


Figure 7: Activation vectors obtained from models trained on 2 randomized classes.

classifying novel data, applications of which can be used to tackle tough text classification problems in domains like forensic linguistics.

Our future work will involve improving the number and diversity of classifiers used in the ensemble. In addition, we plan to consider different neural network architectures that learn sequential information from text, namely variants of *recurrent neural networks* like *Long Short-Term Memory* networks with *attention* mechanism.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Nos. IIS-1359275 and IIS-1659788. We are thankful for the support of BML Munjal University, particularly Prof. Sudip Sanyal and Dr. Satyendr Singh. We also thank Diptodip Deb and Kyle Yee for their insightful discussions and constant encouragement.

References

Abhijit Bendale and Terrance Boult. 2015. Towards open world recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1893–1902.

Abhijit Bendale and Terrance E Boult. 2016. Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1563–1572.

Enrique Castillo. 2012. *Extreme value theory in engineering*. Elsevier.

Bhavana Dalvi, William W Cohen, and Jamie Callan. 2013. Exploratory learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 128–143. Springer.

Laurens De Haan and Ana Ferreira. 2007. *Extreme value theory: an introduction*. Springer Science & Business Media.

Tri Doan and Jugal Kalita. 2017. Overcoming the challenge for text classification in the open world. In *Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual*, pages 1–7. IEEE.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Geli Fei and Bing Liu. 2015. Social media text classification under negative covariate shift. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2347–2356.

Geli Fei and Bing Liu. 2016. Breaking the closed world assumption in text classification. In *HLT-NAACL*, pages 506–514.

ZongYuan Ge, Sergey Demyanov, Zetao Chen, and Rahil Garnavi. 2017. Generative openmax for multi-class open set classification. *arXiv preprint arXiv:1707.07418*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Lalit P Jain, Walter J Scheirer, and Terrance E Boult. 2014. Multi-class open set recognition using probability of inclusion. In *European Conference on Computer Vision*, pages 393–409. Springer.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Fayin Li and Harry Wechsler. 2005. Open set face recognition using transduction. *IEEE transactions on pattern analysis and machine intelligence*, 27(11):1686–1697.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 413–422. IEEE.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Madison, WI.

Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. 2013. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2624–2637.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Ajita Rattani, Walter J Scheirer, and Arun Ross. 2015. Open set fingerprint spoof detection across novel fabrication materials. *IEEE Transactions on Information Forensics and Security*, 10(11):2447–2460.

Joseph John Rocchio. 1971. Relevance feedback in information retrieval. *The Smart retrieval system-experiments in automatic document processing*.

Walter J Scheirer, Anderson Rocha, Ross J Micheals, and Terrance E Boult. 2011. Meta-recognition: The theory and practice of recognition score analysis. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1689–1695.

Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. 2013. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772.

Walter J. Scheirer, Lalit P. Jain, and Terrance E. Boult. 2014. Probability models for open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 36, November.

M Nawaz Sharif and M Nazrul Islam. 1980. The weibull distribution as a general model for forecasting technological change. *Technological Forecasting and Social Change*, 18(3):247–256.

Noam Slonim and Naftali Tishby. 2000. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 208–215. ACM.

Richard L Smith. 1990. Extreme value theory. *Handbook of applicable mathematics*, 7:437–471.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Predicting User Competence from Linguistic Data

Yonas Demeke Woldemariam
Dep. Computing Science
Umeå University
Sweden
yonasd@cs.umu.se

Suna Bensch
Dep. Computing Science
Umeå University
Sweden
suna@cs.umu.se

Henrik Björklund
Dep. Computing Science
Umeå University
Sweden
henrikb@cs.umu.se

Abstract

We investigate the problem of predicting the competence of users of the crowd-sourcing platform Zooniverse by analyzing their chat texts. Zooniverse is an online platform where objects of different types are displayed to volunteer users to classify. Our research focuses on the Zoonivers Galaxy Zoo project, where users classify the images of galaxies and discuss their classifications in text. We apply natural language processing methods to extract linguistic features including syntactic categories, bag-of-words, and punctuation marks. We trained three supervised machine-learning classifiers on the resulting dataset: k-nearest neighbors, decision trees (with gradient boosting) and naive Bayes. They are evaluated (regarding accuracy and F-measure) with two different but related domain datasets. The performance of the classifiers varies across the feature set configurations designed during the training phase. A challenging part of this research is to compute the competence of the users without ground truth data available. We implemented a tool that estimates the proficiency of users and annotates their text with computed competence. Our evaluation results show that the trained classifier models give results that are significantly better than chance and can be deployed for other crowd-sourcing projects as well.

1 Introduction

The science crowd sourcing platform Zooniverse hosts a large number of different projects where volunteers/users (in this paper, the term "volunteer" is used interchangeably with "user") help scientists by classifying various kinds of data. In order to make the experience as positive as possible for the volunteers, so that they are more likely to stay on and contribute to the projects, the Zooniverse team is very interested in anything that can help them understand their volunteers better.

In this article, we explore how much the text comments left by volunteers in the chat rooms accompanying the project Galaxy Zoo can help us in determining their level of proficiency or competence in classifying images. Proficiency is only one among many interesting qualities, and the text data is only one tool for measuring it. The output from the machine learning algorithms we use can be combined with other measures to learn more about user proficiency. Here, though, we focus on the following main question: Does the linguistic data from the chats contain useful information about the volunteers, in particular about the quality of their classifications?

The reason for focusing on Galaxy Zoo, rather than one of the many other projects run by Zooniverse, is that it is one of the oldest and largest projects, which means that there is quite a lot of data available – many users, many classifications, many text comments.

There are several challenges that have to be addressed when trying to answer our question. The hardest one is how to measure the quality of users' classifications. The problem is that there is no ground truth data available. For most of the galaxy photos that volunteers have classified, we do not know the correct answer. No expert in the field has studied and classified them, since the whole point of using volunteers is that the experts do not have the time to do so.

Our approach to this challenge is to use majority votes, i.e., we consider the answer to a question given by the majority of the users to be the correct one. This is by no means an unobjectionable assumption. We describe our approach in more

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 476–484

detail and provide some justification for it in Section 3.

Once a quality measure for each user that has also provided sufficiently many textual comments has been computed, we employ three different machine learning algorithms to the data in order to see whether the values can be predicted from text. Each algorithm is tested on six different sets of features of the textual data. The algorithms we use are k-Nearest Neighbors, Naive Bayesian Classification, and Decision Trees (with gradient boosting).

The results achieved are not spectacular, but they show that analysis of the textual data gives a significantly better than chance prediction of the quality of a users classifications. As mention above, this can be combined with other measures to get better predictions.

To investigate how well our methods generalize to other settings we also test them on data from the Zooniverse Snapshot Serengeti project. The results are encouraging in that they are comparable to the results for Galaxy Zoo.

We discuss related work in Section 2, the calculation of majority votes in Section 3, the experimental setup in Section 4, the experimental results in Section 5 and, finally, the discussion in Section 6.

2 Related work

In the literature a users' competence refers to various kinds of competence. Automated essay scoring, for instance, assesses an author's writing competence or capabilities by analyzing the author's text. An author's competence can also refer to competence or expertise in a specific topic that he/she demonstrates by, for example, his/her written argumentation in a chat discussing the topic. An author's competence can also be related to the author's competence in performing a specific task (e.g. classifying galaxy images) and the author's written text about the task performance can be used to investigate whether there exist correlations. We are interested in the correlation between an author's task performance competence (i.e. correct classification of galaxy images) and his/her chat entries, where the text in the chat entries is not necessarily about the task at hand.

Researchers have intensively investigated methods for automated essay scoring by statistical analysis of linguistic features extracted from text. Automated essay scoring is the process of automatically analyzing text and grading it according to some predefined evaluation criteria. In McNamara et al. (2008), for instance, the authors investigate to what degree high- and low-proficiency essays can be predicted by linguistic features including syntactic complexity (e.g. number of words before the main verb). Their results indicate that high-proficiency writers use a more complex syntax in terms of the mean number of higher level constituents per word and the number of words before the main verb, than low-proficiency writers. In addition, the results indicate that high-proficiency writers use words that occur less frequently in language. Chen and He (2013) improve automated essay scoring by incorporating the agreement between human and machine raters. The feature set to indicate essay quality includes lexical, syntactic, and fluency features. The syntactic features include sentence length, the mean number of subclauses in each sentence, the sum of the depth of all nodes in a parse tree as well as the height of the parse tree. In Pérez et al. (2005), students' essays are assessed by combining an algorithm that includes syntactic analysis and latent semantic analysis.

Linguistic features in written text (e.g. chat) have also been used to predict how competent the authors are with respect to learning and understanding discussed chat topics. Dascalu et al. (2014), for instance, assess the competences of chat participants. To this end, they consider the number of characters written by a chat user, speech acts, keywords and the topics. In addition, social factors are taken into account. The authors generate a social network graph that represents participants' behaviors and participants can be characterized as knowledgeable, gregarious or passive. The social network is used to compute metrics such as closeness, graph centrality, betweenness, stress, and eigenvector.

Linguistic features have been used to predict text-specific attributes (e.g. quality of text) as well as author-specific attributes. In Kucukyilmaz et al. (2008) the authors predict user-specific and message-specific attributes with supervised classification techniques for extracting information from chat messages. User-specific attributes include, for example, gender, age, educational background, income, nationality, profession, psychological status, or race. In Kucukyilmaz et al.

(2008) a term-based approach is used to investigate the user and message attributes in the context of vocabulary use and a style-based approach is used to investigate the chat messages according to the variations in the authors' writing styles.

Another kind of author-specific attribute is the self-confidence of an author. In Fu et al. (2017) the authors investigate how confidence and competence of discussion participants effect the dynamics and outcomes of group discussions. The results show that more confident participants have a larger impact on the group's decision and that the language they use is more predictive of their confidence level than of their competence level. The authors use bag of words, number of introduced ideas, use of hedges (i.e. expressions of uncertainty or lack of commitment) and expressions of agreement as indicators for confidence.

Berry and Broadbent (1984) investigate the relationship between task performance and the explicit and reportable knowledge about the task performance (i.e. concurrent verbalization). The results indicate that practice significantly improves task performance but has no effect on the ability to answer related questions. Verbal instructions of how to do the task significantly improves the ability to answer questions but has no effect on task performance. Verbal instructions combined with concurrent verbalization does lead to a significant improvement in task performance, whereas verbalization alone has no effect on task performance or question answering. The authors Berry and Broadbent (1984) use statistical comparisons of questionnaires.

In Chen et al. (2014), the authors use machine learning techniques (e.g. logistic regression, SVM) to assesss medical students' competencies in six geriatric competency domains (i.e. medication management, cognitive and behavioral disorders, falls, self-care capacity, palliative care, hospital care for elders). The medical students' clinical notes are analyzed and the used linguistic features include bag of words, concepts, negation and semantic type. The authors also use non-linguistic features such as the number of clinical notes for the competence assessment.

3 Computing majority votes

Schwamb et al. (2005) assess how competently a volunteer can identify planetary transits in images. This is done within the Planet Hunter project[1] which is a crowd sourcing project for which volunteers classify planet images. A decision tree helps volunteers in identifying light curves in the images and the volunteers then mark transit features visible in the light curve which results in a so-called transit box. The classifications are stored in a database and for each entry question in the decision tree, the time stamp, user identification, light curve identifier, and response are stored. In addition, the position of the transit box center, its width and height are stored. As a gold standard synthetic transit light curves are used (i.e. labelled images) where these synthetic transits are mixed into the images that are not labelled for the volunteers to classify. In order to identify the most competent volunteers a weight is assigned based on their tendency to agree with the majority opinion and in case they classified synthetic light curves on their performance of identifying transit events. The user weights' are assigned in two stages. First, all users start out equal and then the results of identifying the synthetic light curves are used to obtain an initial weighting. For every synthetic light curve and volunteer classifier it is evaluated how well the user identified the transit events. If a volunteer identified transits correctly her weight is increased and if a volunteer did not mark any synthetic transits (transit box) her weight is decreased. For all the volunteers who classified non-synthetic images the competence evaluation is based on majority opinion. A volunteer's weight increases if the volunteer is in line with the majority weighted vote and is decreased if the volunteer is not in line with the majority opinion.

One of the major obstacles to our investigation was that there is no gold standard data available for the Galaxy Zoo subjects. (A subject is the Zooniverse term for a unit that is presented to volunteers for classification. In the case of Galaxy Zoo, this is one photo taken by a telescope.) In other words, we do not know what the correct classification for the images are. This, in turn, means that there is no way of computing a gold standard for the competence level of the volunteers, since we cannot with certainty determine whether they have classified an image correctly or not.

For these reasons, we had to find a way of estimating the competence levels. How best to do this is not at all obvious. The one approach that

[1] planethunters.org

we have judged possible is to use majority votes, in essence trusting that most classifications are correct. This assumption is at least in part justified by the fact that if it were not true, the whole Galaxy Zoo project would be pointless. The lack of gold standard data prevented us from using a more sophisticated model, where the volunteers performance on classification tasks with a known answer is used as an initial weighting, which is then reinforced by considering majorities on other classification tasks. Such an approach has been used in Planet Hunters, another Zooniverse project (Schwamb et al. (2005)).

In order to explain our approach in detail, we must first say something about the structure of the classification tasks the volunteers are presented with. Each subject is associated with a decision tree based flow chart of questions. The exact chart varies slightly depending on which sub-project of Galaxy Zoo the subject belongs to, but generally, the volunteers are asked three to five questions for each subject, where each of the questions following the first one depends on the answers to the previous questions. Since most subjects in the database have between 10 and 20 classifications, we determined that computing the majority votes for a whole subject classification, including all the questions from the flow chart, would not be advisable, since the answers to the questions after the first one vary to a surprising degree. We thus made the pragmatic decision to only consider the answers to the first question for each subject.

When a volunteer is presented with a subject, the first question, irrespective of which sub-project the subject belongs to, is whether the object in the middle of the photo is a smooth galaxy, a galaxy with features (a disc, spiral arms, etc.), or looks like a star or some other artifact. There are thus three possible answers to the first question. The first step was therefore to calculate, for each subject, how many volunteers had given answers 1, 2, and 3, respectively. In order to have a reasonable amount of data for each subject, we disregard subjects with fewer than 10 classifications.

The next step was computing a competence value for each volunteer that had done at least 10 classifications. Here, we again had some design choices to make. The easiest approach would have been to simply say that for each subject, the correct answer is the one that has gotten the most votes, and then count, for each volunteer, how many times they had given the correct answer and dividing this number by the number of classifications the volunteer had performed. This approach, however, has serious drawbacks. In the data set, it is not uncommon to find subjects where no answer has a clear majority. Consider a case where answer 1 has 12 votes, answer 2 has 10, and answer 3 has 4. Here, it is not clear which answer is actually correct, and it would be bad to give a "full score" to the volunteers that had given answer 1 and no points at all to those that had given answer 2.

Instead, we decided to go by the assumption that the more other volunteers agree with you, the more reasonable your answer is. We thus calculated the competence score for a volunteer as follows. For each subject that the volunteer has classified, we divide the number of votes that agree with the volunteer by the total number of votes, getting a number in the interval $[0, 1]$. The score for the volunteer is then the average of these numbers over all subjects the volunteer has classified. This approach has the benefit of "punishing" a volunteer more severely for an incorrect answer to an "easy" question, where most other volunteers have voted for another answer, while being lenient towards false answers to "hard" questions. On the downside, the users answering the hard questions correctly, get less credit for this than they deserve.

4 Experimental setup

4.1 Text Analysis and Feature Extraction

We extracted text comments written by 7,839 volunteer. We only targeted those users who classified at least 10 subjects and discussed at least one of their classifications in chat text. The users were divided into three categories of equal size based on their computed competence levels on a scale ranging from 0 to 1: low ($[0, 0.52]$), medium ($(0.52, 0.59]$) and high ($(0.59, 1]$). Having an equal number of users in each category helps to achieve balanced data and in eliminating bias during the machine learning phase. The raw data was obtained from Zooniverse Galaxy Zoo as a database dump. The entire text data contains around 26,617 sentences with average sentence length of 5.02. We extracted three types of linguistic features out of the text data: bag-of-words, syntactic and punctuation marks. The number of classifications is also included in each feature set as special feature or meta data.

4.1.1 Syntactic feature set

To extract syntactic features the Stanford probabilistic context-free grammar (PCFG) parser was used Klein and Manning (2003). These features provide a lot of information about the complexity of the syntactic structures used by the volonteers. For each syntactic category, we made a correlation analysis with classification competence. To this end, we implemented a Java-based program that reads user texts from the database stored on the Mongodb server running on a local machine and makes use of the PCFG model to construct a syntactic parse or phrase-structured tree for the texts. The program counts the frequency of syntactic categories/constituent tags occurring within the tree and then annotates the text with these tag count information.

The non-leaf nodes in the resulting tree has three major syntactical categories: lexical categories, functional categories and phrasal categories. The lexical categories are the part-of-speech tags of the leaf nodes that represent content words that make up the parsed text, for example, NN (noun), JJ (adjective), VB (verb), etc. As the Stanford parser has been trained on the Penn Treebank, we use the part-of-speech tags and their notations used in the tree bank to label the non-leaf nodes as well as to identify categories. The functional categories contain items responsible for linking syntactic units, for example, DT (determiner), IN (preposition), MD (modal), etc. The phrasal categories represent different type of phrases within a sentence for which the parse tree is built, for example, NP (noun phrase), VP (verb phrase) and AP (adjective phrase), etc. In the syntactic feature set there are 66 numerical attributes representing the frequency count of syntactic categories.

We attempted to analyze the correlation between the syntactic categories count with computed competence values by looking at the correlation coefficient(CC) calculated for each syntactic category as summarized and shown in Figure 1. The calculated CC values range $[-0.05, 0.04]$, statistically speaking these values do not show that there is a strong relationship. The Figure basically shows three types of relationships between the syntactic categories and competence according to the observed CC values: the first type of relationship is exhibited by the categories laid over the left-hand side of the X-axis such as JJR (adjective, comparative), PRP$ (possessive pronoun) and JJS (adjective, superlative) they are negatively correlated with competence, those concentrated around the center such as S (simple declarative clause), PRT (particle) and WP$ (possessive wh-pronoun), do not seem to have a correlation with competence and the third type of relationship is exhibited by the categories close to the right-hand side of the X-axis such as PRP (personal pronoun), SQ (inverted yes/no question) and SBARQ (direct question introduced by a wh-word).

4.1.2 Punctuation mark feature set

We also extracted the frequency count of punctuation marks including question mark, period, and exclamation mark. Special characters such as # and @ were also included. We also tried to perform a correctional analysis between each feature in the set with competence as we did for the syntactic feature set and we got quite similar results in terms of the strength of their relationship. In the punctuation mark feature set there are 7 numerical attributes, that correspond to the selected punctuation marks.

4.1.3 Bag-of-Words feature set

We used the text analysis package of Rapidminer[2] and text-processing Java libraries to extract the Bag-of-Words (BoW) and punctuation marks features respectively. The text analysis involves splitting text into sentences, each sentence is further split into words followed by stemming and part-of-speech tagging. In the Bag-of-Words feature set there are 19,689 attributes excluding the target (label) attribute, i.e competence. Each attribute has a numerical value that represents the frequency count of any token in a text.

By taking both an individual feature set and combination of them, we came up with 6 feature set configurations: Bag-of-Words (BoW),punctuation marks (Pun), punctuation marks with Bag-of-Words (Pun+BoW), syntactic, syntactic with Bag-of-Words (Syn+BoW), and the combination of BoW, punctuation mark and syntactic (BoW+Pun+Syn).

4.2 Training, Validation and Evaluation

We trained and evaluated three machine learning classifiers: Decision Trees (DT) with gradient boosting, Naive Bayes (NB) and k-Nearest Neighbor (KNN). These three methods were also used in

[2] `rapidminer.com`

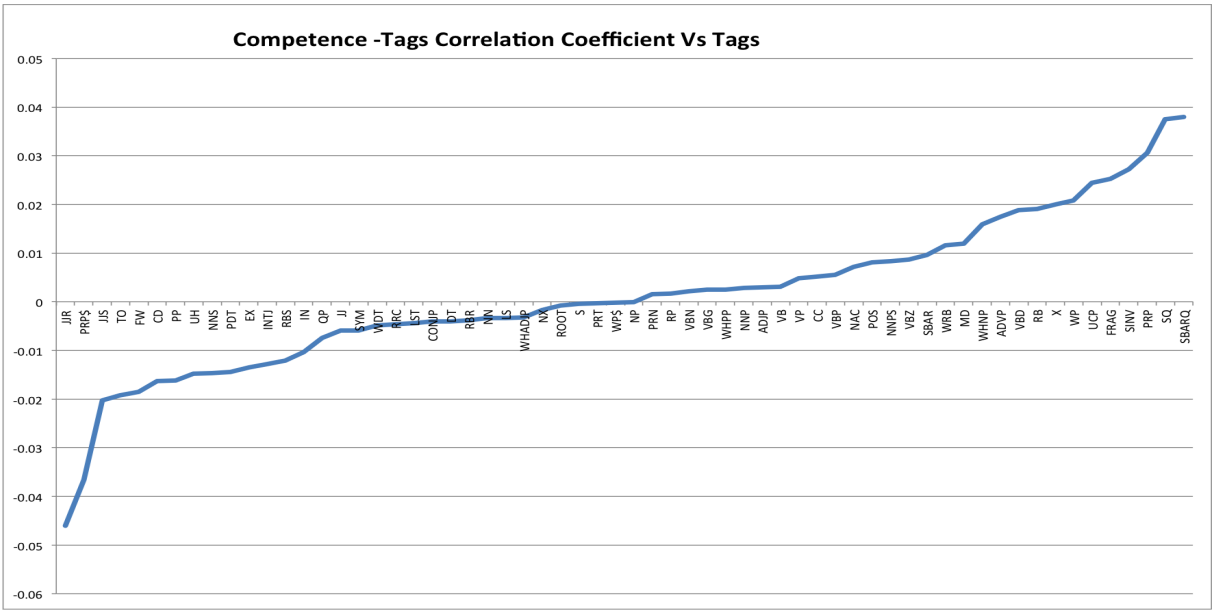


Figure 1: The correlation between frequency of the extracted syntactic categories and computed competence values

a previous study Woldemariam (2017) using Snapshot Serengeti data (another Zooniverse project). As the implementation of these classifiers is available in Rapidminer Studio, we trained them on the Galaxy Zoo data set after configuring the model parameters associated with each classifier.

We adopted the best practices of the machine learning life cycle that includes randomly sampling and dividing the data into a training set, a validation (development) set and a test (evaluation) set, deciding the size of each set and balancing the proportion of examples in each class of users. According to this, the classifiers are trained on 80% of the entire text corpus with the selected feature sets. The remaining 20% is used to evaluate the trained models. We set aside 10% of the training set as a development data set to optimize model parameters.

4.2.1 Training

The classifiers were trained with the different feature sets. The feature sets are applied for each classifier as shown and denoted in the Table 1, first, Bag-of-Words (BoW), second, punctuation marks (Pun), third, punctuation marks with Bag-of-Words (Pun+BoW), fourth, syntactic, fifth, syntactic with Bag-of-Words (Syn+BoW), and sixth, the combination of BoW, punctuation mark and syntactic (BoW+Pun+Syn). Each classifier is trained 6 times with these 6 feature set configurations. Thus, in total, 18 (3*6) classifiers models are produced for the subsequent validation phase. The training set contains texts from 6,262 unique users.

4.2.2 Validation

As a part of the training phase, we attempted to answer whether the trained classifiers are statistically significant before we evaluate them. We performed a null-hypothesis (H_0) test, aiming at checking that the prediction made by the models is not likely just by random chance. In the null-hypothesis we assume that the mean accuracy value before and after testing the models is the same. However, in principle any effective model must have a greater mean accuracy after the testing and reject H_0.

In statistics there are different ways of testing the null hypothesis and the most widely used approach for machine-learning problems associated with models significance test is a T-test. Basically, there are two important parameters in the T-test, a t-value and a p-value. The t-value indicates that how far the mean of the test sample is from the known mean (μ_0), for example, the accuracy mean before testing a model, depends on the size(n), mean ($\bar{x}$) and the standard deviation(s) of a test sample as shown in the Equation 1. The p-value shows how likely the two means are to be equal. Once the t-value is calculated, the p-value can be obtained from a T-table by using degrees of freedom (df).

$$t = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n}}} \quad (1)$$

So, we performed the t-test for each model with the development set. We found that all the models scored a p-value below 0.001.

4.2.3 Evaluation

The models were evaluated with two equal size test sets by using accuracy and F-measure metrics. The first set is from the same domain as the training set, and the second one is from the Zooniverse Snapshot Serengeti forum discussion posts.

To be able to use the Snapshot Serengeti data, we had to overcome the mismatch of the intervals of the competence scales of the two domains. We had to use a strategy that allows adapting the way that the competence scale for the Galaxy Zoo is divided to label its users to the Snapshot Serengeti users. In Woldemariam (2017), there are two scales used to divide the Snapshot Serengeti users, the first scale divides the user into three groups (Low, Medium and High) and the calibrated scale divides the users into five groups (very Low, Low, Medium, High, very High). Thus, we decided to use the first scale, as it is closer to the Galaxy Zoo scale in terms of the number of divisions, though the intervals between the groups are not exactly the same.

5 Results

The results of the trained classifiers on the test sets are summarized in Table 1. We consider two performance metrics: accuracy and F-measure. To calculate accuracy we take the fraction of true positive and true negative instances (correctly classified instances) among the test instances, while the overall F-measure is computed by macro-averaging the F-measure values over classes. That means the harmonic mean of precision and recall of each class, i.e. the local F-measure of each class, is calculated, then we take the average value over classes as an overall F-measure.

The first thing to notice is that the accuracy scores are low. Since there are three classes in our data (Low, Medium, and High), a random classifier would be expected to have an accuracy of 33%. In our tests, the best classifiers achieve an accuracy of just over 40%. There are, however, reasons why this is not as negative a result as it might seem. First, we work with relatively little data, since most Galaxy Zoo users do not write comments, and no gold standard data is available. There is therefore reason to hope that the approach would yield better results in similar settings, but where more data is available. Second, for the intended use case, Zooniverse, any result that is statistically certain to be better than random is useful. Zooniverse needs a better understanding of their volunteers, both when evaluating the results from classification tasks and in order to learn how to encourage and educate the volunteers. Our classification methods can be combined with other user data to generate such knowledge.

Another interesting aspect is that the results for Snapshot Serengeti are not significantly worse than those for Galaxy Zoo, which indicates that the approach generalizes and can be deployed for other projects as well.

Analyzing the data in more detail, the k-Nearest Neighbors (KNN) classifier performs best overall and in particular when syntax is not involved. When using syntax, it is slightly worse and is sometimes outperformed by the Decision Trees (DT) classifier. It is also interesting that on the Galaxy Zoo data, the best performance (KNN on BoW and PunMM and DT on Syn) are seen when classifiers use only one of the three feature sets. Combining the sets seem to muddy the waters. A partial explanation for this could be that BoW has so many more features than the other two sets.

We also note that the performance of DT and KNN are so similar that we cannot, based on our tests, confidently say that one is a better choice than the other for this task.

The Naive Bayesian (NB) classifiers generally performed the poorest. One potential reason for this is that KNN and DT have flexible model parameters, such as k for KNN and the number and depth of the trees for DT. These values were noted to greatly impact the prediction accuracy during the validation phase. For example, by varying the value of k of KNN model, we achieved about 5% increase in accuracy. Varying the values of the parameters of the kernel-based NB did not help very much in the improvement of its performance.

We also observe that Punctuation mark (PunM) feature set gives the best accuracy value of 40.32% and F-measure value of 40.05%, in this case the Galaxy Zoo test set is used. Generally, according to the evaluation and comparison performed on this test set, the feature sets or their combinations

Table 1: Models Evaluation and Comparison Results, the **All(3)** column is equivalent with BoW+PunMM+Syn

Metric	Domain	Classifier	Feature set					
			BoW	PunMM	PunMM+BoW	Syn	Syn+BoW	All(3)
Accuracy (in %)	Galaxy Zoo	DT	39.55	39.49	38.85	**39.74**	**39.55**	**39.55**
		NB	38.08	37.64	37.32	38.27	38.27	38.27
		KNN	**40.06**	**40.32**	**40.00**	39.30	39.23	39.11
	Snapshot Serengeti	DT	39.30	38.66	39.04	38.85	39.30	**40.19**
		NB	37.70	37.44	37.83	37.64	37.64	37.96
		KNN	**40.26**	**39.94**	**39.74**	**40.26**	**40.19**	39.87
F-measure (in %)	Galaxy Zoo	DT	38.79	39.17	38.25	**39.37**	**38.79**	**38.79**
		NB	37.36	36.76	34.87	37.49	37.49	37.49
		KNN	**39.85**	**40.05**	**39.68**	38.74	38.68	38.47
	Snapshot Serengeti	DT	34.42	36.89	**38.19**	35.21	34.42	30.53
		NB	37.68	**37.61**	37.61	**37.63**	**37.63**	**38.10**
		KNN	**38.08**	37.16	36.87	37.45	37.41	36.72

used in study can be put in this order based on their relative influence on the prediction of competence from text: PunMM, BoW, PunMM+BoW, Syn, Syn+BoW or BoW+PunMM+Syn. The ranking changes a bit when the Snapshot Serengeti test set is used for the evaluation i.e. BoW, Syn, Syn+ BoW or BoW+PunMM+Syn, PunM, PunM+BoW. This ranking style compares the feature sets based on their impact on a single best classifier among the three (DT, KNN and NB). There are other ways of ranking the feature sets that consider the average performance of all the three instead concerning both accuracy and F-measure.

We also tried to analyze how the Punctuation mark, the syntactic features and their combination affect of the performance of the classifiers over the Bag of Words features. Regardless the domains of the test sets involved in the evaluations, we observe that the performance of NB (BoW based) is improved by adding syntactic and punctuation marks features. Likewise, the DT (BoW based) is affected by adding syntactic and the combination of syntactic and punctuation mark features.

6 Discussion

The approaches used in this study, from user competence calculation to machine learning tasks, can be improved or possibly yield different results with alternative strategies proposed in the following paragraphs.

The most obvious approach is to use data labeled by domain experts. For Galaxy Zoo, such data is not available, but we could consider other possibilities, such as a semi-supervised bootstrapping method if we had a small amount of labeled data. Semi-supervised bootstrapping methods have been effective in various text analysis problems, such as topic and sentiment-based text classification Zhou et al. (2013). In competence estimation, to reduce dependency on majority voting, we train a classifier on a small dataset labeled by experts sampled from the training corpus. We then use the classifier to label the remaining unlabeled samples in the training corpus and retrain the classifier iteratively until we reach certain stop criteria.

Feature wise, in addition to the selected feature sets, we could use more features such as universal dependencies, character n-gram, bag-of-topics. The syntactic feature set extracted can be further enriched with features extracted using a dependency parsing to describe and represent the syntactic structure of users text better. Dependency parsing captures the dependency relationships between syntactic units/words and has been used to improve the accuracy of text classification tasks Nastase et al. (2006). As a part of improving our research results, we have also carried out preliminary experiments on a character n-gram and bag-of-topic features, where we describe a user text with topic words extracted using a topic modeling technique. We found that both types of features improve the accuracy of the trained models to a certain degree.

Finally, using multiple metadata information about users from other external data sources, for example, capturing their participations in either other seasons of the Galaxy Zoo project or other projects of Zooniverse, may help to better model the users competence.

References

D. C. Berry and D. E. Broadbent. 1984. On the relationship between task performance and associated verbalizable knowledge. *The Quarterly Journal of Experimental Psychology*, Section A. 36(2):209–231.

H. Chen and B. He. 2013. Automated essay scoring by maximizing human-machine agreement. *In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*, 1741–1752.

Y. Chen, J. Wrenn, H. Xu, A. Spickard, R. Habermann, J. Powers, and J. D. Denny. 2014. Automated assessment of medical students? clinical exposures according to aamc geriatric competencies. *In AMIA Annual Symposium Proceedings Archive*, 375–384.

M. Dascalu, E-V. Chioasca, and S. Trausan-Matu. 2008. ASAP–an advanced system for assessing chat participants. *In AIMSA: International Conference on Artificial Intelligence: Methodology, Systems and Applications*, volume 5253 of Lecture Notes in Computer Science. Springer. 58–68.

Y. Woldemariam 2017. Predicting competence from text. *In Proceedings of The 21st World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI)*, 147–152.

L. Fu, L. Lee, and C. Danescu-Niculescu-Mizil. 2008. When confidence and competence collide: Effects on online decision-making discussions. *In Proceedings of the 26th International Conference on World Wide Web, WWW ?17*, 1381–1390.

D. Klein and C.D. Manning. 2003. Accurate unlexicalized parsing. *In Proceedings of the 41st Meeting of the Association for Computational Linguistics*, 423–430.

T. Kucukyilmaz, B. Cambazoglu, C. Aykanat, and F. Can. 2008. Predicting user and message attributes in computer-mediated communication. *Information Processing and Management*, 44(4):1448–1466.

D.S. McNamara, S.A. Crossley, and P. M. McCarthy. 2010. Linguistic features of writing quality. *Written Communication*, 27(1):57–86.

D. Pérez, A. M. Gliozzo, C. Strapparava, E. Alfonseca, P. Rodríguez, and B. Magnini. 2005. Automatic assessment of students? free-text answers underpinned by the combination of a bleu-inspired algorithm and latent semantic analysis. *In Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference*, Clearwater Beach, Florida, USA, 358–363.

M. E. Schwamb, C. J. Lintott, D. A. Fischer, M. J. Giguere, S. Lynn, A. M. Smith, J. M. Brewer, M. Parrish, K. Schawinski, and R. J. Simpson. 2012. Planet hunters: Assessing the kepler inventory of short-period planets. *The Astrophysical Journal*, 754(2):129.

V. Nastase, J. Shirabad, and M. Caropreso. 2006. Using Dependency Relations for Text Classification. *Proceedings of the 19th Canadian Conference on Artificial Intelligence*, 12–25

Y.Haralambous, Y. Elidrissi, and P. Lenca. 2014. Arabic Language Text Classification Using Dependency Syntax-Based Feature Selection. *Proceedings of the 19th Canadian Conference on Artificial Intelligence*

G. Zhou, J. Li, D. Zhao, and Y. Feng. 2013. Semi-supervised Text Categorization by Considering Sufficiency and Diversity . *Natural Language Processing and Chinese Computing.*, 105–115.

Neural Morphological Disambiguation Using Surface and Contextual Morphological Awareness

Akhilesh Sudhakar
IIT (BHU), Varanasi, India
akhileshs.s4@gmail.com

Anil Kumar Singh
IIT (BHU), Varanasi, India
nlprnd@gmail.com

Abstract

Morphological disambiguation, particularly for morphologically rich languages, is a crucial step in many NLP tasks. Morphological analyzers provide multiple analyses of a word, only one of which is true in context. We present a language-agnostic deep neural system for morphological disambiguation, with experiments on Hindi. We achieve accuracies of around 95.22% without the use of any language-specific features or heuristics, which outperforms the existing state of the art. One contribution through this work is building the first morphological disambiguation system for Hindi. We also show that using phonological features can improve performance. On using phonological features and pre-trained word vectors, we report an accuracy of 97.02% for Hindi.

1 Introduction

Morphologically inflected words are derived from a root by modifying it (e.g., by applying prefixes, suffixes and infixes) based on linguistic features (manifested as the inflection tagset). Morphological analysis involves extracting this root word and the set of features that describe the inflected form. These analyses contain syntactic and semantic information about inflected words. Table 1 shows an example for the Hindi word 'पूरे' [pUre][1]. Existing morphological analyzers typically work in isolation, meaning that they generate multiple analyses of a word, purely based on surface structure. For many NLP tasks like machine translation and topic modeling, however, it is essential to know which morphological analysis is correct in the context of the sentence. Morphological disambiguation aims to solve this problem. The task of disambiguation is non-trivial and is complicated by the dependencies of the correct analysis on the surface structure of the inflected word, on the surface structures of the neighboring words, and on the analyses of neighboring words.

We present a deep neural morphological disambiguation system that leverages context information as well as surface structure. While we have experimented on Hindi in our work, we report accuracies without employing any language-specific features to show that our system can generalize across different languages. We also show performance boost by using phonological features and pre-training of word vectors. To the best of our knowledge, this is the first ever non-naive morphological disambiguation system to be built for Hindi.

Like other Indo-Aryan languages, Hindi is morphologically rich and a word form may have over 40 morphological analyses (Goyal and Lehal, 2008). Though the inflectional morphology of Hindi is not agglutinative, the derivational suffixes are. This leads to an explosion in the number of inflectional root forms (Singh et al., 2013). One of the reasons for our focus on Hindi is that it has a wide coverage of speaking population, with over 260 million speakers across 5 countries[2] and is the fifth most spoken language in the world[3]. We present four neural architectures for this task, each different from the others by the nature of context information used as disambiguating

[1] We use the Roman notation popularly known as WX for representing Hindi words

[2] https://www.ethnologue.com/statistics/size

[3] The exact rank may be a matter of debate due to the socio-linguistic scenario in South Asia, with some surveys claiming it to be even more popularly spoken.

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 485–494

Root	Category	Gender	Number	Person	Case	TAM	Suffix
pUrA	adj	m	sg	-	o	-	-
pUrA	adj	m	pl	-	d	-	-
pUrA	adj	m	pl	-	o	-	-
pUrA	n	m	pl	3	d	0	0
pUrA	n	m	sg	3	o	0	0
pUra	v	any	sg	2	-	ए	e
pUra	v	any	sg	3	-	ए	e
pUra	v	m	pl	any	-	या	yA

Table 1: Morphological analyses of the word `पूरे' [pUre] (A '-' indicates that the feature is not applicable and an 'any' indicates that it can take any value in the domain for that feature)

evidence. We assess our results by implementing an existing state-of-the-art system (Shen et al., 2016) on our Hindi dataset. Our system outperforms this state-of-the-art system.

2 Related Work

There is very little directly corresponding previous work on morphological disambiguation and it cannot be formulated in the same way as POS tagging. This is because the number of classes is fixed in POS tagging, whereas it is variable in our problem. Still, since part of speech (POS) tagging is a closely related task, the work on POS tagging can also provide useful insights. However, morphological disambiguation is a harder task to perform than POS tagging. The earliest approaches to POS tagging were rule-based (Karlsson et al. (1995), Brill (1992)) and required a set of hand-crafted rules learnt from a tagged corpus. More recently, Kessikbayeva and Cicekli (2016) present a morphological disambiguation system using rules based on disambiguations of context words.

Statistical approaches are also used for morphological disambiguation. Hakkani-Tür et al. (2000) propose a model based on joint conditional probabilities of the root and tags. Sak et al. (2007) use a perceptron model, while other statistical models use decision trees as by Görgün and Yildiz (2011). Hybrid approaches have also been tried, with Orosz and Novák (2013) using an approach combining rule-based and statistical approaches, to prune grammar-violating parses.

The use of deep learning for morphological disambiguation, has been explored. Straka and Straková (2017) build a neural system for tasks such as sentence segmentation, tokenization and POS tagging. Plank et al. (2016) build a multilingual neural POS tagger. While we draw insights from works on tasks such as POS tagging, we bear in mind that POS tagging and morphological disambiguation are significantly different. Morphological disambiguation is more complex because it works with multiple categories and not just part-of-speech. This introduces sparseness in the model, as well as considerations of whether the different categories are dependent on each other, on how to combine classifiers for each category, etc. The number of analyses for a word also varies.

Yildiz et al. (2016) propose a convolutional neural net architecture, which takes context disambiguation into account. Shen et al. (2016) use a deep neural model with character-level and well as tag-level LSTMs to embed analyses. Our work shares certain aspects in common with theirs but is different in many ways. We experiment on Hindi (which has significantly different morphological properties from the three languages they explore), use different neural structures, show the effect of language specific phonological features and study the impact of unsupervised pre-training of embeddings under different settings. Further, as mentioned earlier, we consider their results to be state-of-the-art because theirs is a language-agnostic system which gives state-of-the-art results on all 3 languages they have experimented on. We show that our model gives better performance than an implementation of their best model on Hindi.

3 Neural Models

We present four models for morphological disambiguation. Some aspects are common among them. They all use a deep neural network, which, given the current word in con-

sideration and one of the candidate morphological analyses of the word, acts as a binary true/false classifier. A final softmax layer outputs probabilities for correct and incorrect, based on whether the candidate analysis is correct or not. An ideal classifier would predict the probability of correct as 1 and incorrect as 0 for the correct morphological analysis of the word. As is usual in word sense disambiguation, we make the 'one sense per collocation' assumption (a word in a particular context has only one correct morphological analysis), with which our dataset is in accordance. The choices of neural architectures used by us are influenced by the findings in the work of Heigold et al. (2016), in which the authors conclude that on morphological tagging tasks, different neural architectures (CNNs, RNNs etc.) give comparable results, and careful tuning of model structure and hyperparameters can give substantial gains. We also draw insights from their work on augmenting character and word-level embeddings.

3.1 Terminology Used

For each of 'category', 'gender', 'number', 'person', 'case', 'TAM' and 'suffix', we use the term ***'feature'***. We call each of the values of a feature for a particular word, a ***'tag'***. For instance the feature 'gender' can have tags 'M (male)', 'F (female)' and 'N (neuter)'. The root and the tagset together make up a morphological analysis for a word. We use the term ***'candidate analysis'*** to refer to each of the morphological analyses generated by the analyzer for a given word.

3.2 Broad Basis for the Architectures

We first establish an intuitive and statistical foundation to justify our decision choices in building the deep neural network. We extract dependencies between roots and features from the work by Hakkani-Tür et al. (2000), noting that the assumptions used for Turkish by the authors hold good for Hindi too. We also obtain surface-information-related dependencies from the work by Faruqui et al. (2016). The following is the full set of extended dependencies:

- **Dependency #1**: The root of a word depends on the roots as well as the features of all previous and following words in the window
- **Dependency #2**: Each feature of a word depends on the roots and the features of all previous and following words in the window
- **Dependency #3**: Each feature of a word depends on the root of the current word, as well as all other features of the current word
- **Dependency #4**: The root and each feature of a word depend on the surface form of the word
- **Dependency #5**: The root and each feature of a word depend on the surface forms of the word as well as those of all previous and following words in the window.

In all these four models, the following network components are also consistent.

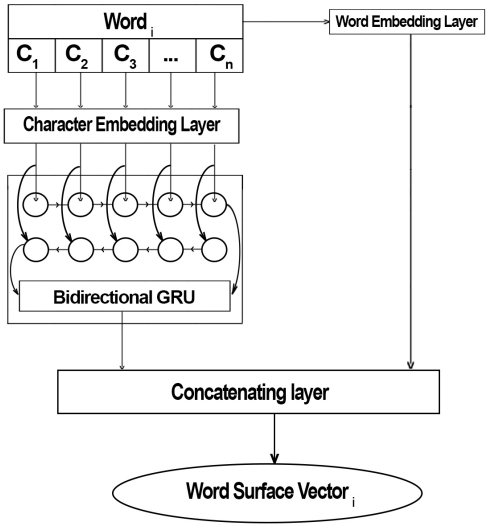


Figure 1: Architecture for the word surface vector. 'i' indicates the i^{th} input word.

3.3 Word Input

Word inputs to the network are embedded at two levels. A word embedding vector is generated using the word as a whole. Each character in the word is also embedded in a character embedding vector and these character embeddings are fed, in sequence, to a bidirectional GRU. The output vector of the GRU and the

word embedding vector are concatenated together to form the ***'word surface vector'*** that takes into account surface features of the word. The part of the network that generates the word surface vector is shown in Figure 1.

The character-level GRU allows for capturing of surface properties of a word, and takes into account Dependency #4 (section 3.2). We obtain marginal accuracy gains (of around 0.1%) by using a GRU instead of an LSTM at the character-level.

3.4 Candidate Analysis Input

Candidate analysis inputs to the network are treated as two inputs: the root word and rest of the tags. The root is treated in the same exact fashion as the word inputs (for the same reasons mentioned in the above section) with the only difference being that all words share a common embedding layer, while all roots share a separate common embedding layer. Consequently, a corresponding ***'root surface vector'*** will be generated as described for each root input. Tagsets are represented as binary vectors, with positional encoding. The root surface vector and all the tag encodings are treated as a sequence and fed as input to a bidirectional LSTM. This design choice, including the bidirectionality, has been made to address Dependency #3 (section 3.2). We call the output vector of this LSTM, the ***'root features sequence vector'***. The part of the network architecture that generates the root features sequence vector is shown in Figure 2.

3.5 Hyperparameters and Training

All GRUs and LSTMs have a hidden layer size of 256, and deep GRUs and LSTMs have a depth of 2 layers. Deep convolutional networks have a filter width of 3, hidden layer size of 64 and depth of 3 layers. The model can run for 10,000 epochs but we make use of early stopping with a patience of 10 epochs in order to keep the generalization error in check. This is a validation-based early stopping on the development set. The word and root embedding layers have a dimension of 100, while the character embedding layer has a dimension of 64. All models use the categorical cross-entropy loss function and the Adam optimization method as proposed by Kingma and Ba (2014). The sequence of words in each

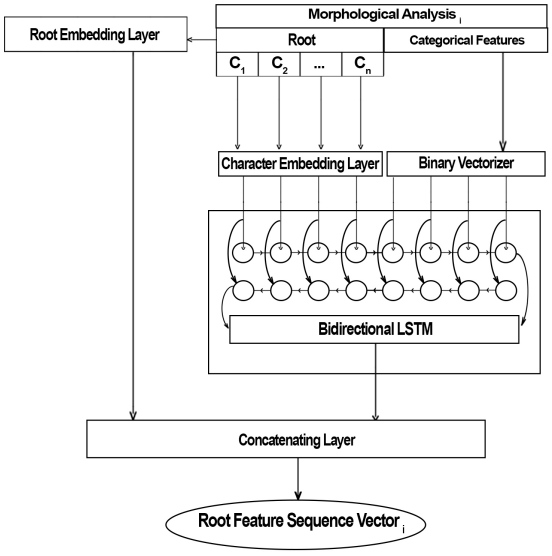


Figure 2: Network architecture for generating the root feature sequence vector. The subscript i indicates the morphological features of the i^{th} input word. These might be candidate analyses or correct analyses, depending on where (training or testing) they are used in other figures.

sentence act as a mini-batch during training. The best model is saved for predictions on the test data.

3.6 Baseline Model

As mentioned earlier, since there does not exist a non-naive state of the art system for Hindi, we use a low baseline model that picks one candidate analysis at random and predicts this to be the correct morphological analysis for the given word. This[4] is the default for building several machine translation (MT) systems, such as the Sampark[5] system, for Indian languages. The most that is currently done for these MT systems is to apply some agreement based rules[6]. Proper evaluation of these modules may be needed, but it is beyond the scope of this paper. It must be mentioned here that the baseline we have picked is a weak baseline. However, we have done so for a couple of reasons. Firstly, we compare our results to the state-of-the-art system mentioned earlier and show performance gain. Therefore it is not a case of inflation of results using a weak

[4]The so called 'pick one morph' module.
[5]`http://sampark.org.in`
[6]The 'guess morph' module.

baseline. Secondly, we have presented 4 models which show a gradation of performance on this task (as shown in Table 5). Thirdly, the baseline presents the case when absolutely no character, word or context-level information is available to the model. We believe that since we study the impact of each of these kinds of knowledge on the models' performance, we must also study the case when none of this knowledge is available.

3.7 Model 1

This model solely relies on the surface information of the current word to make predictions about its morphological analysis. Given a current word and the root and tags of the candidate analysis, the model uses only the word surface vector (section 4.2) with the root features sequence vector (section 4.3) of the candidate to make predictions. Figure 3 shows the exact structure of the network used.

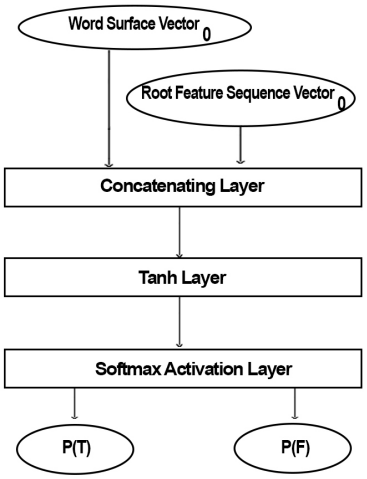


Figure 3: Structure of Model 1. The subscript '0' indicates the current word. P(T) and P(F) indicate probability of True and False respectively.

3.8 Model 2

This model makes use of not only the current word's surface information but also the surface information of all words in a window which has 4 words to the left and 4 words to the right of the current word in the sentence. (We experiment with values from 2 to 6). Building the model this way accounts for Dependency #5 (section 3.2). This model uses out-of-sentence tokens too, to ensure that words towards the beginning and end of the sentence also have a full window. The word surface vectors (section 4.2) of all the words in this window, along with the root features sequence vector (section 4.3) of the candidate are fed into a bidirectional LSTM in this model. Figure 4 shows the exact network structure used.

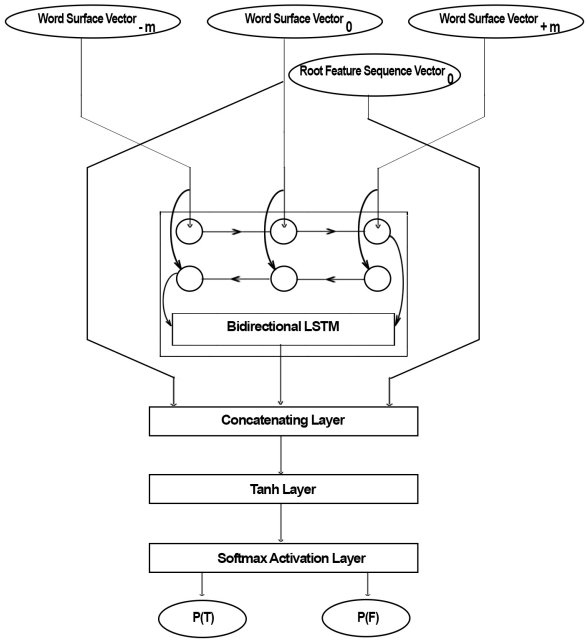


Figure 4: Model 2. The subscript '0' indicates the current word, the subscript '-m' represents any word in the left context of the current word and the subscript '+m' represents any word in the right context of the current word. P(T) and P(F) indicate probability of True and False respectively.

3.9 Model 3

This model makes use of not only the current word's surface information and the surface information of all words in a window which has 5 words to the left of the current word in the sentence, but also the correct morphological annotations of the words in this left-context. This model partially accounts for Dependency #1 and Dependency #2 (section 3.2). The word surface vector (section 4.2) of each word is concatenated with the root features sequence vector (section 4.3) of the word to give a 'complete vector'. Complete vectors, each concatenated with their own convolutions are fed as inputs to a deep LSTM. Model 3 uses the network structure shown in Figure 5, except that Figure 5 shows the model using

the current word's left and right context, while Model 3 uses only its left context.

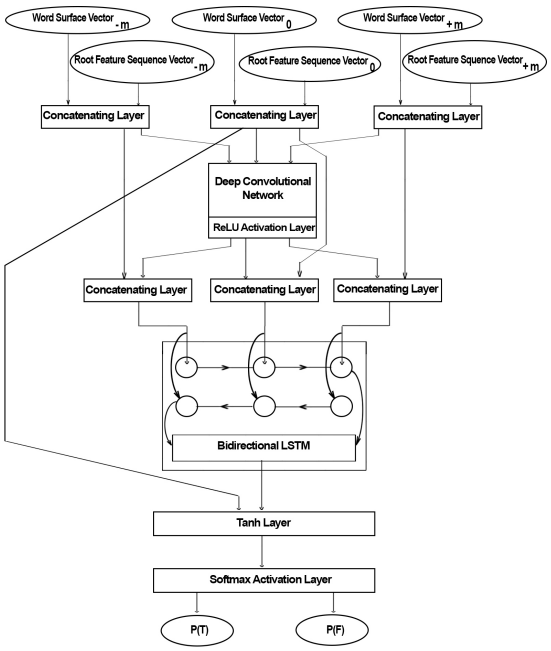


Figure 5: Model 4. Model 3 also has a similar configuration except for the context of the current word. The subscript '0' indicates the current word, the subscript '-m' represents any word in the left context of the current word and the subscript '+m' represents any word in the right context of the current word. P(T) and P(F) indicate probability of True and False respectively.

3.10 Model 4

Model 4 is similar to Model 3, except that this model makes use of surface information and correct morphological annotations of not only words to the left but also to the right of the current word. The complete window for the current word has 4 words to the left and 4 words to the right of it. (We experiment with values from 2 to 6). This model takes into account all the dependencies mentioned in section 3.2. Figure 5 shows the network structure for Model 4.

It must be mentioned here that Model 3 and Model 4 are slightly complex due to the presence of a CNN and further concatenation of the convolved inputs with the original inputs. We have conducted experiments on simpler versions of Model 3 and Model 4 but with poorer results (an average drop in accuracy of 1.2%). An elaborate discussion of these results is not possible due to space constraints. Specifically, we used the same context as used in these two models but did not use the intermediary CNN in these simpler experiments. The reasons for the performance improvement upon using a CNN could be that CNNs have proved to be particularly useful for classification tasks on data that has the property of local consistency. This is evident from previous work on using CNNs for similar classification tasks such as those by Collobert et al. (2011), Yildiz et al. (2016) and Heigold et al. (2016). Well-formed sentences of any language (Hindi, in our case) display local consistency because they have a natural order, with context words forming abstractive concepts/features. Hence, a CNN was an intuitive choice for our task.

4 Experimental setup

4.1 Dataset

We use a manually annotated Hindi Dependency TreeBank[7], which is part of the Hindi-Urdu Dependency TreeBank (HUTB)[8] as the source of the correct morphological analysis of words in the context of their sentences. The treebank annotates words from sentences taken from news articles and textual conversations. Each word in every sentence of the treebank is annotated with the correct morphological analysis. We use a morphological analyzer for Hindi[9], which was developed earlier, but is now used for the Sampark MT system and other purposes. We use it to generate the different possible morphological analyses for each word (in isolation) in the treebank. For each word in the treebank, the candidate analysis matching the word's treebank-annotated morphological analysis is labeled as true while all other candidate analyses are labeled as false. In the entire dataset, we ensure that out of all the candidate analyses of a word, there is one that matches the treebank annotation for that word. Table 2 provides specific statistics about the dataset used. Table 3 describes the features of a morphological analysis as well as provides the domain of possible tags (values) for each feature.

[7] http://ltrc.iiit.ac.in/treebank_H2014
[8] http://verbs.colorado.edu/hindiurdu
[9] http://sampark.iiit.ac.in/hindimorph

Attribute	Count
Total words	298,285
Unique words	17,315
Manual additions of treebank annotation	115432
Ambiguous words	179,453
Unambiguous words	118,742
Sentences in treebank	13,933
Mean sentence length	21.40
Mean morphological analyses per word	2.534
Mean morphological analyses per ambiguous word	3.550
Standard deviation of morpho--logical analyses per word	1.620
Maximum morphological analyses for a word	10

Table 2: Dataset statistics

Feature name	List of possible tags
Root	Not fixed
Category	Noun(n), Pronoun(pn), Adjective(adj) verb(v), adverb(adv) post-position(psp), avvya(avy)
Gender	Masculine(m), Feminine(f), Neuter(n)
Number	Singular(sg), Plural(pl), Dual(d)
Person	1st Person(1), 2nd Person(2), 3rd Person(3)
Case	Direct(d), Oblique(o)
TAM	है,का,ना,में,या,या1,से,ए, कर,ता,0,था,को,गा,ने
Suffix	kA,e,wA,WA,yA,nA,ko,ne, 0,kara,gA,yA1,meM,se,hE

Table 3: Domain of tags (values) for each feature. The tag 'TAM' denotes the tense, aspect and modality marker.

We would like to mention here that since our system is built to pick the correct analysis from the morphological analyses generated by the analyzer, it assumes that every word has a set of candidate analyses. In the context of our task, it is not relevant to discuss the case when the morphological analyzer itself fails to provide candidate analyses.

Table 4 shows the number of ambiguous words and the total number of words used for training, development and testing. The test set is held-out and is used solely for reporting final results. The development set is used to validate and tune model hyperparameters. During testing, the model is provided with only the different candidate morphological analysis outputs from the morphological analyzer, for each word. The correct analyses (also referred to as annotations in the following section) provided by the treebank for each word are used to calculate the reported accuracies.

Phase	Ambiguous Words	Total Words
Training	149,540	248,572
Development	11,963	19,885
Testing	17,950	29,828

Table 4: Word counts in training, development and test splits

4.2 Methods of Testing

Models that use morphological analyses of the context words (Models 3 and 4) have access to correct annotations of these contexts during training and validation but not during testing. During testing, in order to provide 'correct annotations' of words in the context of a word, to these models, we use Model 2 or Model 3. This is because Model 2 does not itself use context annotations and Model 3 itself uses only left context annotations (it can hence, predict the correct morphological analysis for each word in the test data from left to right, treating the predictions of the previous words as the correct annotations of the left-context of the words that occur next). However, in the case where Model 3 is being used as the test set annotator, a strategic choice has to be made for annotation. A greedy strategy would pick the morphological analysis with the highest softmax probability of being the correct annotation and annotate the word with this annotation. However, the greedy strategy fails if the model makes mistakes towards the start of a sentence or performs poorly on only some types of words, because these errors propagate to every consecutive word and get compounded. In order to avoid these kinds of errors, we use a beam search with width 10 for pre-annotating the test set in the case of context-based models.

5 Results and Analysis

Table 5 presents performance accuracies of different models and with different methods used to annotate test data in the cases where an initial pre-annotation of test data is needed, as discussed in the previous section. As mentioned before, model accuracy is calculated by comparing each trained model's predictions on

Model	Pre-testing Test Data Annotation Model	Accuracy on Ambiguous Words	Accuracy on All Words
Baseline	NA	29.40	38.43
S-O-T-A	S-O-T-A	90.13	92.06
Model 1	NA	80.21	83.96
Model 2	NA	87.35	89.18
Model 3	Treebank	93.82	96.17
Model 3	Model 2	89.40	93.35
Model 3	Model 3	91.23	94.90
Model 4	Treebank	**94.77**	**97.59**
Model 4	Model 2	90.41	94.74
Model 4	Model 3	92.65	95.22

Table 5: Performance of different models (all accuracies are percentages). S-O-T-A stands for state-of-the-art, which is the full-context model of Shen et al. (2016). The accuracy gain we have achieved over S-O-T-A is 2.8% on ambiguous words and 3.43% on all words.

the test data with the correct analyses from the treebank data, regardless of which model was used for the initial test annotation (if any). The standard measure for accuracy is used:

$$\frac{\#\ of\ correct\ disambiguations}{total\ \#\ of\ words\ in\ test\ set}$$

For practical purposes, the best performing system is the last row in Table 5, i.e., in which Model 4 uses Model 3 for pre-annotating the test set (though the 7th row has the highest accuracy, we cannot assume treebank annotations on the test data as well). Table 5 also presents the results of using the existing state of the art (S-O-T-A) model on our Hindi dataset. We have used the best performing model (on our Hindi dataset) proposed by Shen et al. (2016), the full-context model, as the state of the art. The accuracy gain we have achieved over state of the art is 2.8% on ambiguous words and 3.43% on all words.

We suggest possible reasons for the observed performance behavior in table 5. Typologically, Hindi is a Subject-Object-Verb, head-final language and uses post-positional case marking. This means that on an average, words show disambiguation dependencies on the words following them. However, there is also disambiguation evidence for a word to be gained from its left context. For instance, adverbs usually occur before (to the left of) the verb or object they refer to. Similarly, relative clauses, adjectives and articles are written before the noun they refer to. Model 4 uses the morphological analyses of the right-context of a word as well as the left context and hence is able to leverage information from both preceding and following words. Hence it is able to achieve better performance than Model 3. Models 2 and 1 do not leverage evidence about the morphological analysis of the words in the window and perform worse than the other two models. This shows (as is also quite intuitive) that the morphological analysis of the context is far stronger evidence in disambiguating a word, than just the surface forms of the words and its context. Model 2 performs better than Model 1 as it has access to the surface forms of the surrounding words, which in turn provide some level of knowledge about their inflected properties.

From control experiments, we conclude that our gain over the state of the art is due to factors that include careful tuning of hyper-parameters, increasing model complexity and leveraging the strength of combining models. At the end of section 3.10, we have already described the advantage of using a CNN. The existing state of the art does not leverage this advantage. Further, in allowing Model 3 to pre-annotate the test data, we have allowed our full-context model to take advantage of the strengths of a left-to-right model, which is also something that the existing state of the art does not explore.

5.1 Language-specific Enhancements

While the reported results in Table 5 are obtained without using pre-training of word vectors or phonological features, we also experimented with using these enhancements. We present results on the experimental setup where we train using Model 4 and pre-annotate the test set using Model 3. All performance improvements are reported as those obtained over and above the performance of this particular experiment setup.

5.1.1 Pre-training of Word Vectors

We pre-trained word embeddings using the word vector representation methods proposed by Bojanowski et al. (2016). This method makes use of an unsupervised skip-gram model to generate word vectors of dimension 100. We used an augmented corpus comprising of Wikipedia text dump for Hindi, as well as

Model	Accuracy	Accuracy gain over Baseline (%)
Baseline	38.43	0
Model 4	95.22	147.74
Model 4 + Pre-training	96.64	151.47
Model 4 + Phonological Features	96.04	149.91
Model 4 + Pre-training + Phonological Features	**97.02**	**152.46**

Table 6: Performance with language-specific enhancements

the collection of news articles and conversations that the Hindi treebank annotated words come from. Using vector pre-training gave us an accuracy improvement of 1.42%. One of the main reasons for the performance boost obtained during pre-training could perhaps be that the pre-trained word vectors capture syntactic and morphological information from short neighbouring windows.

5.1.2 Use of Phonological Features

Morphology interacts closely with phonology and there is ample work on the phonology-morphology interface (Booij, 2007). It is quite intuitive, therefore, to use phonological features (Chomsky and Halle, 1968) for a morphological problem. Besides, Hindi is written in the Devanagari script, in which the mapping from letters to phonemes is almost one to one. Each letter can therefore be represented as a set of feature-value pairs, where the features are phonological features such as type (whether consonant or vowel), place, manner etc. (Singh, 2006). This is true for almost all languages that use Brahmi-derived scripts. Phonological features are incorporated into the model by concatenating them with the character-level embeddings for words. We observe a performance enhancement of 0.82% upon using these phonological features.

Employing pre-training as well as phonological features boosted our model's performance from 95.22 % to 97.02%. These enhanced results are summarized in Table 6.

6 Future Work

We plan to test all our models on different languages and analyze which models perform best on each language and hope to be able to correlate these results with the linguistic phono-morphological properties of the languages. We will also try out this model in the Sampark[10] machine translation system to evaluate the effect it has on translation.

Recently, an attention-based machine translation model was proposed by Bahdanau et al. (2014) that defines a selective context around a word rather than a fixed window for all words. Models 3 and 4 can be modified to use an attentional mechanism based on the context words' positional and morphological properties. This would allow these models to increase their range of information-capturing across words in the sentence, without losing information due to propagation in a recurrent unit running across a large window. Experiments have been done in the past for morphological disambiguation using Conditional Random Fields (CRFs). It might be interesting to see the hybrid use of CRF models with the models we propose.

7 Conclusion

We propose multiple deep learning models for morphological disambiguation. We show that the model that makes use of morphological information in both the left and right context of a word performs best on this task, at least in the case of Hindi. We also study the effect of different context settings on model performance. The differences in performance obtained using these different context settings, we believe, follows from the typological and morphological properties of the language. Hence, we also believe that different languages may work better with different models that we propose. The use of phonological features enhances the quality of predictions by these models, at least in the case of Hindi.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vec-

[10] `https://sampark.iiit.ac.in/sampark/web/index.php`

tors with subword information. *arXiv preprint arXiv:1607.04606*.

Geert Booij. 2007. *The interface between morphology and phonology*. Oxford University Press.

Eric Brill. 1992. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, ANLC '92, pages 152–155, Stroudsburg, PA, USA. Association for Computational Linguistics.

Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English*. Harper & Row, New York.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proc. of NAACL*.

Onur Görgün and Olcay Taner Yildiz. 2011. A novel approach to morphological disambiguation for turkish. In *Computer and Information Sciences II*, pages 77–83. Springer.

Vishal Goyal and Gurpreet Singh Lehal. 2008. Hindi morphological analyzer and generator. In *Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference on*, pages 1156–1159. IEEE.

Dilek Z Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2000. Statistical morphological disambiguation for agglutinative languages. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 285–291. Association for Computational Linguistics.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2016. Neural morphological tagging from characters for morphologically rich languages. *arXiv preprint arXiv:1606.06640*.

Fred Karlsson, Atro Voutilainen, Juha Heikkila, and Arto Anttila, editors. 1995. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Walter de Gruyter & Co., Hawthorne, NJ, USA.

Gulshat Kessikbayeva and Ilyas Cicekli. 2016. A rule based morphological analyzer and a morphological disambiguator for kazakh language. *Linguistics and Literature Studies*, 4(1):96–104.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

György Orosz and Attila Novák. 2013. Purepos 2.0: a hybrid tool for morphological disambiguation. In *RANLP*, volume 13, pages 539–545.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529*.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2007. Morphological disambiguation of turkish text with perceptron algorithm. *Computational Linguistics and Intelligent Text Processing*, pages 107–118.

Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell, and Chris Dyer. 2016. The role of context in neural morphological disambiguation. In *COLING*, pages 181–191.

Pawan Deep Singh, Archana Kore, Rekha Sugandhi, Gaurav Arya, and Sneha Jadhav. 2013. Hindi morphological analysis and inflection generator for english to hindi translation. *International Journal of Engineering and Innovative Technology (IJEIT)*, pages 256–259.

Anil Kumar Singh. 2006. A computational phonetic model for indian language scripts. In *Constraints on Spelling Changes: Fifth International Workshop on Writing Systems. Nijmegen*, Nijmegen, The Netherlands.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99.

Eray Yildiz, Caglar Tirkaz, H. Bahadir Sahin, Mustafa Tolga Eren, and Ozan Sonmez. 2016. A morphology-aware network for morphological disambiguation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2863–2869. AAAI Press.

WORD SENSE DISAMBIGUATION FOR MALAYALAM IN A CONDITIONAL RANDOM FIELD FRAMEWORK

Junaida M K
IT Education Centre
Thalassery Campus
Kannur University
junaidashukoor@gmail.com

Jisha P Jayan
Centre for Development of Imaging Technology
Thiruvanathapuram
jishapjayan@gmail.com

Elizabeth Sherly
Indian Institute of Information Technology and Management-Kerala
Thiruvanathapuram
sherly@iiitmk.ac.in

Abstract

Word Sense Disambiguation (WSD) or Lexical Ambiguity Resolution is one of the pressing problems in Natural Language Processing (NLP), which identifies the correct sense of an ambiguous word in the specific context in a given sentence. WSD is considered as a harder problem as it depends on a set of classes, which vary depending on the context. This paper describes two algorithms Conditional Random Field (CRF) and Margin Infused Relaxed (MIRA) in a CRF framework for Malayalam WSD. This framework makes use of the contextual feature information along with the parts of speech tag feature in order to predict the various WSD classes. For training set, number of ambiguous words has been annotated with 25 WSD classes. The experimental results of the 10 fold cross validation shows the appropriateness of the proposed CRF based Malayalam word sense tagger.

1 Introduction

Word Sense Disambiguation is an intermediate task which is necessary at one level or another to accomplish in most natural language processing tasks. The development of an automatic Word Sense Disambiguation requires either a sense inventory, usually obtained from a dictionary, thesaurus or a large annotated corpus. The significant amount of information about the word and its neighbours of a particular word in a context gives a sense of a particular word which can be useful in a language model for different speech and text processing applications. In English 'line' (cord, division, formation, phone, product, text), 'hard' (difficult to achieve, intense, intense, for surfaces, things), interest (stake, involvement, interestingness, pastime, the thing that is important, charge for borrowing money) are the words with multiple meanings and such words are called polysemy.

WSD is a major subtask of Machine Translation, have relevant significance in almost every application of language technology, including information retrieval, lexicography, knowledge mining/acquisition and semantic interpretation, and is becoming increasingly important in new research fields such as the cognitive Computing, semantic web, bioinformatics etc.

Automatic WSD systems are available for many languages like English, Spanish, Chinese and some Indian languages. Malayalam being an unstructured language, faces a severe problem in the work on automatic WSD. Malayalam is an agglutinating language that exhibits very rich system of morphology and many senses, which is challenging. The basic components required for developing good WSD is the availability of Malayalam dictionary/thesaurus and labelled text corpus. For example,Consider the following sentence from Malayalam: ഞാൻ ആവന്റെ കരം പിടിച്ചു (njan avanTe karaM piTiccu) with the meaning I took his hand and രാമു അവന്റെ വീടിന്റെ കരം അടച്ചു (raamu avanTe veeTinTe karaM aTaccu) with the meaning ramu paid his tax.The word കരം (karaM) have different meanings Tax or Hand. Here the distinction of the sense of the word കരം (karaM) is complex due to the lack of capitalization information and free word order of the language.

This paper is organized into different sections. First section dealt with the introduction part. The second section explains the major works carried out in this area. The third and fourth sections describe the complexity of Malayalam language and the Machine learning approach using CRF framework based on two algorithms CRF and MIRA. The next two sections explain the proposed work and implementation. The seventh section includes the experimental results obtained. The eighth sec-

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 495–502

tion concludes the paper with future works that can be done as an outcome of this work.

2 Related Works

There are many approaches used for identifying WSD. The two main approaches are Dictionary based and Corpus based. The Dictionary-based method, uses external knowledge resources, which define explicit sense distinctions for assigning the correct sense of a word in context. In corpus-based methods, machine-learning techniques are used to induce models of word usages from large collections of text examples. Both knowledge-based and corpus -based methods have their own benefits and drawbacks. The former approach mainly uses external lexical resources like dictionaries, thesaurus, WordNet etc. They are easy to implement as it requires only simple look up of knowledge resources like machine readable dictionary. The corpus based methods use techniques from statistics and machine learning to induce language models. Learning can be done with supervised or unsupervised methods, which learns sense classifiers from annotated data with minimal or partial human supervision respectively. Many standard machine learning techniques have been applied, including Naive Bayes (NB), Maximum Entropy (ME), Exemplar-based (kNN), Decision Lists (DL), Support Vector Machines (SVM) etc. Naive Bayes algorithm is one of the simplest algorithm, which uses Bayes rule and given the class labels conditional independence of the features are assumed. It has been applied to many experiments in Natural Language Processing as well as WSD with considerable success (Yuret, 2004).

The information theory in particular,The Maximum Entropy approach provides a flexible way to combine statistical evidences from many sources. It has been applied to many NLP problems and also appears as alternative in WSD (Suarez, 2002). Chatterjee (2009) presented a trainable model applies the information theory for Word Sense Disambiguation (WSD) for resolving the ambiguity of English words. Decision Lists were used for lexical ambiguity resolution in Spanish and French accent restoration (Yarowsky, 1994) and used in other work for WSD (Yarowsky, 1995). Parameswarappa (2011) described the machine learning techniques with naive bayes classifier for Kannada target word sense disambiguation using compound words clue and syntactic features in a local context.

Lesk (1986) was one of the first researchers who tried to disambiguate Machine Readable Dictionaries (MRD) using Simplified Lesk algorithms. His algorithm became well-known among WSD researchers. His algorithm was primarily an overlap based algorithm which suffers from overlap scarcity. These methods, highly rely on lexical resources such as machine readable dictionaries, thesaurus etc. For English, this method achieved 50-70% accuracy in correctly disambiguating the words. The work (Sinhar, 2004) mainly focused on Hindi. They used contextual overlap between sentential context and extended sense definitions from Hindi Word Net. Sense bag was created by extracting words from synonyms, glosses, example sentences, hyponyms, and glosses of hyponyms, example sentences of hyponyms, hypernyms, and glosses of hypernyms, example sentences of hypernyms, meronyms, glosses of meronyms, and example sentences of meronyms. A context bag was created by extracting words in the neighborhood i.e. one sentence before and after, of the polysemous word to be disambiguated. The sense which maximized the overlap was assigned as winner sense. By using word co- occurrences of the gloss and the context (Gaona, 2009) presented a measure for sense assignment useful for the simple Lesk algorithm. Based on domain information and WordNet hierarchy (Kolte, 2009) proposed unsupervised approach to WSD. The words in the sentence contribute to determine the domain of the sentence. The availability of WordNet domains makes the domain-oriented text analysis possible. The domain of the target word can be fixed based on the domains of the content words in the local context. This approach can be effectively used to disambiguate nouns.

In Malayalam only a few works have been published. A knowledge based approach to Malayalam WSD (Haroon, 2010) has been done. It is based on a hand devised knowledge source and uses the Lesks and Walkers algorithm and also using the concept of conceptual density with Malayalam WordNet as the Lexical resource. The knowledge based system will result in poor accuracies because of the dependency of the algorithm on the stored tag words within the knowledge source.

3 Complexity of Malayalam

This section introduces the linguistic preliminaries of Malayalam language and complexities involved in the Malayalam Word Sense Disambiguation. The world languages are classified into fixed word order and free word order. In fixed word order the words constituting a sentence can be positioned in a sentence according to grammatical rules in some standard ways. On the other hand, in the free word order no fixed ordering is imposed on the sequence of words in a sentence. The English language is example of fixed word order language and Sanskrit is pure free word order language. Generally Malayalam is a free word order language and agglutinating language and exhibits very rich system of morphology. Morphology includes inflection, conflation (sandhi), and derivation. Word Sense Disambiguation is a difficult task in Natural Language Processing, In addition to the difficulties involved in Word Sense Disambiguation, the complexity level is even more in unstructured language like Malayalam. Here we will briefly describe the complexities involved in our work. For example, consider a sentence അവൻ നടന്നു With the meaning He walk and യോഗം നടന്നു With the meaning Meeting executed. Here the distinction of the sense of the word നട is very complex due to the lacks of capitalization information and free word order of the language. Applying stochastic models to the WSD problem requires large amounts of annotated data in order to achieve reasonable performance. Stochastic models have been applied successfully to English, German and other European languages for which large sets of labeled data are available. The problem remains difficult for Indian languages (ILs) due to the lack of such large annotated corpora. This is due to the fact that many different encoding standards are being used. Also, the number of Malayalam documents are available in the web is comparatively quite limited. Malayalam word sense disambiguation is of interest due to a number of applications like machine translation, text summarization, information retrieval.

To begin with, this experiment requires a sense tagged corpus in -order to achieve considerable accuracy for disambiguation. Developing corpus is a tedious and very time consuming task. The next issue involved in this work is the unavailability of sense inventory which will decide appropriate senses to the specific word in a context. The most appropriate meaning of a word is selected from a predefined set of possibilities, usually known as sense inventories. An efficient POS tagger in Malayalam is required to extract Word Sense Disambiguation, which also requires large corpus for training.

4 Machine Learning Using CRF

Statistical methods work by employing a probabilistic model containing features of the data. Features of the data, that can be understood as rules set for the probabilistic model, are created by learning the resulting corpora with properly marked tags. The probabilistic model then uses the features to calculate and determine the foremost probable tags. As such, if the annotated features of the data are correct and reliable, the model would have a high likelihood to find almost all the tags within a text.

CRF has found its application in many domains that may deal with the structured data. They are considered to be state of the art techniques for many applications in NLP. CRFs are a probabilistic framework (Wallachi, 2004) that is used for labeling and segmenting structured data, such as sequences, trees and lattices. CRFs bring together the best of generative and classification models. These are mainly undirected graphical models (Zhang, 2013). The underlying idea is that of defining a probability distribution which is conditional over label sequences given a particular observation sequence, rather than a joint distribution over both label and observation sequences. A key advantage of CRFs is their great flexibility to include a wide variety of arbitrary, non - independent features of the input (McCallum, 2002). The primary advantage of CRFs over HMMs is their conditional nature, which result in the relaxation of independent assumptions required by HMMs in order to ensure tractable inference. Additionally, CRFs avoid the label bias problem (Lafferty, 2001).

Margin Infused Relaxed Algorithm (MIRA) is a machine learning algorithm for multi-class classification problems. It has been introduced (Crammer, 2003). It learns set of parameters (vector or matrix) by processing all the given training examples one at a time, according to each training example parameters are updated. So that the current training example is classified correctly with a margin against incorrect classifications at least as large as their loss. The change of the parameters

is kept as small as possible. A two-class version called binary MIRA is not requiring the solution of a quadratic programming problem, so it is simple. Binary MIRA can be used in an onevs-all configuration, it can be extended to a multiclass learner that approximates full MIRA, but may be faster to train. The flow of the algorithm looks as follows:

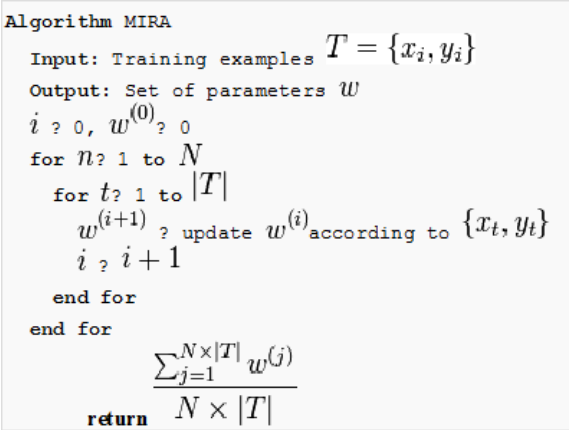


Figure 1: Algorithm MIRA

In the present work, we propose a machine approach using two different algorithms namely CRF and MIRA of Conditional Random Field framework for unrestricted Malayalam text WSD. The main steps involved are corpus collection, preprocessing, tagging, training and analysis. The template for training the CRF engine is defined. A lot of work is being done in the fields of corpus building, creating an efficient POS tagger and subject identification in Malayalam language.

5 Implementation

For WSD implementation, is used CRF++ and the experiment was carried out on different Malayalam ambiguous words. The template file contains the features specified for training and testing. The template file has multiple lines, each corresponds to a particular composite feature. It helps in generating n-gram features from the feature columns. The variables U and B are used to represent the features which denotes uni-gram and bi- gram respectively. The template line that starts with U predicts the current label generating n weights for n different labels. The template line which starts with B defines the current and previous labels generating n*n weights in the model. The composite feature is expressed by %x[i,j] with respect to the current labels. The template for CRF is defined as follows:

```
# Unigram
Unigram U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U05:%x[-1,0]/%x[0,0]
U06:%x[0,0]/%x[1,0]
U10:%x[-2,1]
U11:%x[-1,1]
U12:%x[0,1]
U13:%x[1,1]
U14:%x[2,1]
U15:%x[-2,1]/%x[-1,1]
U16:%x[-1,1]/%x[0,1]
U17:%x[0,1]/%x[1,1]
U18:%x[1,1]/%x[2,1]
U20:%x[-2,1]/%x[-1,1]/%x[0,1]
U21:%x[-1,1]/%x[0,1]/%x[1,1]
U22:%x[0,1]/%x[1,1]/%x[2,1]
# Bigram
B
```

In order to accommodate common words and senses, we have used manually collected sentence from various Malayalam newspapers, Wikipedia articles, blogs, books, novels etc. Table 1 shows these words and sense.

In order to avoid inconsistencies present in spelling, spacing and punctuation, preprocessing is done by thoroughly checking the database. Then manual tagging of polysemous words and parts of speech tagging were carried out.

Feature selection plays an important role in machine learning. The experiments have been carried out using the basic context information and Parts of speech tag combination of word and tag context. The features are binary valued functions which associate a tag with various elements of the context. The experiments used two groups of features: word and word + part-of-speech bigrams. Following are the details of the features that have been applied to WSD task. Word features are lexical features, unique words that occur in the training set in a specific window range. Word feature contains the following attributes. w-2, w-1, w, w+1, w+2, (w- 2,w-1,w), (w-1,w,w+1), (w,w+1,w+2) , where the last three correspond to collocations of three consecutive words. Word + POS features are lexico-syntactic features combining POS information in a predefined range of

Word	Senses (classes)
രസം (rasaM)	താൽപര്യം, കറി, ഇഷ്ടം, രുചി, മെർക്കുറി (taal˜paryaM, kaRi, ishTaM, ruci, mer˜kkuRi)
നട (naTa)	ക്രിയ, നടക്കുക , പടി (kRIya , naTakkuka , paTi)
അടി (aTi)	ചുവടളവ്, പാദം, തല്ല് (cuvaTaLav , paadaM , tall)
വാനം (vaanaM)	മാനം, അഭിമാനം (maanaM , abhimaanaM)
ഉത്തരം (uttaraM)	മറുപടി, ചോദ്യോത്തരം, താങ്ങ് (maRupaTi, coodyoottaraM, taangng)

Table 1: Ambiguous Words and Senses (Classes)

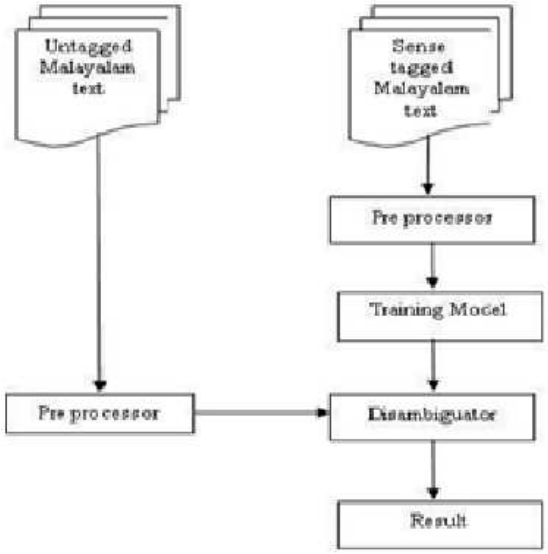


Figure 2: Block Diagram

the particular word. Word + POS feature contains w-2,w-1, w, w+1, w+2, with parts of speech information p, (w,p), (p-1,p). The tagging was performed using BIS tagset. Four taggers have been implemented based on the CRF and MIRA model. The first tagger (Word) makes use of the simple contextual feature, whereas the second tagger (Word+POS) uses parts of speech information features along with the simple contextual features. Each tagger is trained and tested using both the models, CRF based stochastic tagging scheme and MIRA. The same training corpus has been used to estimate the parameters for all the models.

6 Experimental Results

For the evaluation of this experiment we have used n-fold crossvalidation method due to the lack of huge amount of corpus.Usually data is split in to 70% for training and 30% for testing or in some cases 80% for training and 20% for testing. Although this distribution is commonly used for large datasets, it presents a challenge for smaller datasets and it might lead to problem of representativeness of the training or testing data. For these experiments, the method of n-fold cross validation is used divided in ten sets, each set containing 10% of the total data, therefore a ten-fold cross validation. The 10-fold was chosen mainly because the amount of data used for the experiments is not considered to be big as in most other applications. Because of that, fewer partitions were employed in order to ensure that a reasonable number of instances are included in each partition.Therefore it is necessary to ensure that random sampling is done in a way that guarantees that each class in the data set is properly represented in both the training and test sets.

The evaluation of the CRF and MIRA based models has been done using evaluation matrices. We have implemented two CRF and MIRA based models using Word feature and Word + POS feature.The classification is performed for skewed and highly imbalanced data, accuracy is very high and it does not reflect exactly the performance of the classifier. For this reason, precision (P), recall (R) and F-measure (F) scores are reported, which shows how precise and complete the classification is on the positive class. The TP, TN, FP and FN refer to true positives, true negatives, false positives and false negatives respectively. In a binary class based classification context, the terms positive and negative used in these definitions are associated with membership to one of the two semantic classes involved inc the classification (senses).

```
ഉരുക്കിന്റെN_NN      NULL
ആവിർഭാവത്തോടെ V_VM_VNF          NULL
വലിയ     JJ        NULL
വലിയ     JJ        NULL
ഉത്തരങ്ങൾ          N_NN      താങ്ങ്
നിശ്ചയാസം           RB        NULL
നിർമിക്കാമെന്നായി V_VM_VNF          NULL
.         RD_PUNC   NULL
ചോദ്യം    N_NN      NULL
വായിച്ച് V_VM_VNF          NULL
കണ്ണിൽ   N_NN      NULL
ഇരുട്ട്   N_NN      NULL
കയറുമ്പോൾ          N_NNP     NULL
ഉത്തരം    N_NN      ചോദ്യോത്തരം
കിട്ടാതെ  V_VM_VNF          NULL
വട്ടംകറങ്ങുമ്പോൾ V_VM_VNF          NULL
തലയിൽ    N_NN      NULL
മിന്നുന്ന  JJ        NULL
ക്രിയേറ്റിവ്          N_NN      NULL
ഐഡിയാസ്    N_NN      NULL
എഴുതി     V_VM_VNF          NULL
വയ്ക്കാൻ  V_VM_VNF          NULL
കൂടിയുള്ളതല്ലേ      N_NN      NULL
.         RD_PUNC   NULL

ഒരു       QT_QTC    NULL
ഗ്ലാസ്സ്     N_NN      NULL
രസം N_NN  കറി
കുടിക്കാൻ          V_VM_VNF          NULL
കൊതിയാവുന്നു      V_VM_VF   NULL
.         RD_PUNC   NULL

മിനി      N_NN      NULL
സ്കേർട്ടിട്ടാൽ      V_VM_VNF          NULL
മാനം      N_NN      അഭിമാനം
പോകുമോV_VM_VINF          NULL
```

Figure 3: Sample Tagged Corpus

For example, where disambiguation involves the classes മാനം and അഭിമാനം, TP (TN) refers to the മാനം (അഭിമാനം) test occurrences correctly classified as such by the system. Likewise, FP (FN) refers to those അഭിമാനം (മാനം) test occurrences that have been misclassified by the system as belonging to class മാനം (അഭിമാനം).

The experimental results for the 10-fold cross validation test for the CRF-based Malayalam word sense disambiguation system with Word feature and Word+POS feature are presented in 2 and 3 respectively.

The system has demonstrated overall average precision, recall, F- measure values of 58.688, 53.678, and 52.359 respectively for Word Feature. The result shows the overall average precision, recall, F-measure values are 61.387, 49.454, and 51.75 respectively for Word +POS feature.

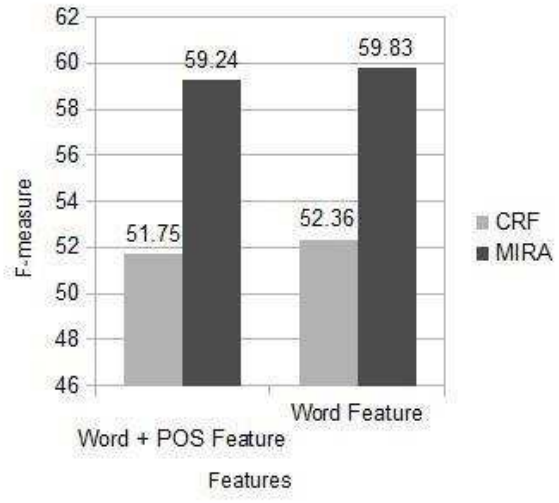


Figure 4: Analysis of F-measure result

The experimental results for the 10-fold cross validation test for the MIRA-based Malayalam word sense disambiguation system with Word feature and Word+POS feature are presented in Table 3. The system has demonstrated overall average precision, recall, F- measure values of 62.598, 63.045, and 59.829 respectively for word feature and 61.387, 49.454, and 59.75 word+POS feature respectively.

The performance evaluation of the models are done using F-measure. Using the value of F-measure the performance result presented in Fig 2 shows that in word feature and word + POS feature, MIRA model outperforms with the CRF model. The use of simple contextual feature give a little improvement for both CRF and MIRA model. Using the F measure, the performance results displayed in the above figure show that regardless of the contextual features or POS information feature the MIRA-based tagger outperforms CRF based framework.

7 Conclusion and Future Directions

7.1 Conclusion

This work addresses CRF based word-sense disambiguation with two different approaches. CRF provides flexibility to include diversity of features. We have used two algorithms in CRF framework which is basic Conditional Random Field algorithm and Margin Infused Relaxed (MIRA) algo-

Test Set	Word Feature			Word + POS Feature		
Sl. no	Precision	Recall	F-measure	Precision	Recall	F-measure
1	57.25	43.74	47.72	58.42	54.2	49.53
2	69.08	56	56.84	67.09	53.99	57.15
3	42.36	46.09	41.29	66.18	50.22	55.13
4	41.34	43.69	40.04	56.69	48.9	49.81
5	68.85	65.27	61.99	61.11	46.2	48.95
6	52.3	56.63	52.22	61.11	50.6	52.88
7	70.05	65.72	62.59	68.39	51.47	55.08
8	66.01	57.69	55.89	68.14	45.83	52.61
9	60.31	52.27	53.94	44.3	40.22	41.47
10	59.33	49.68	51.07	62.44	52.82	54.89
Average	58.68	58.69	52.35	61.387	49.454	51.75

Table 2: RESULTS OF 10 FOLD CROSS VALIDATION USING CRF FOR WORD AND WORD+POS FEATURE

Test Set	Word Feature			Word + POS Feature		
Sl. no	Precision	Recall	F-measure	Precision	Recall	F-measure
1	73.48	71.33	71.05	58.42	54.51	54.88
2	51.9	66.7	56.04	56.37	66.4	57.46
3	73.22	73.71	68.75	60.79	56.07	56.01
4	49.14	54.7	46.13	72.59	63.15	65.55
5	62.1	60.03	58.79	59.49	52.99	52.05
6	67.24	71.2	66.46	70.16	57.22	61.85
7	68.44	58.01	60.62	67.99	57.52	59.8
8	59.5	57.24	55.72	69.05	67.75	65.59
9	70.41	69.83	67.25	62.42	54.87	56.81
10	50.55	47.7	47.48	63.52	63.54	62.44
Average	62.598	63.045	59.829	63.926	59.402	59.244

Table 3: RESULTS OF 10 FOLD CROSS VALIDATION USING MIRA FOR WORD AND WORD+POS FEATURE

rithm. A word sense tagger is created for Malayalam to get an effective word disambiguation using CRF and MIRA. The system is evaluated with manually created words and the accuracy is measured using n-fold cross validation. Results based on the value of F-Measure shows that the performance of MIRA gives the best results with an overall average for word feature precision, recall, F-measure of 62.598, 63.045 and 59.829 respectively for 10-folds. The experimental results are very promising when large amount of annotated corpus was used and handling morphology exhaustively. More words and senses can be added to this so as to increase the accuracy. Other machine learning techniques like Naive Bayes classifier, ME, Neural Networks etc can be applied in this study and the results so obtained can be compared with the existing works.

References

Yuret D. 2004. *Some experiments with a naive bayes wsd system.In Senseval-3: , Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text.* 265-268U.

Suarez A Palomar. 2002. *A maximum entropy-based word sense disambiguation system..*, In*Proceedings of the 19th international conference on Computational linguistics. Association for Computational inguistics* 1 :1-7.

Chatterjee N Misra. 2009. *Word-Sense Disambiguation using maximum entropy model. ,* In*Methods and Models in Computer Science ICM2CS 2009, Proceeding of International Conference on. IEEE.* 1-4.

Yarowsky D. 1994. *Decision lists for lexical ambiguity resolution: application to accent restoration in Spanish and French*, In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics*. 88-95.

Yarowsky D. 1995. . *Unsupervised word sense disambiguation rivaling supervised methods.*, *In Proceedings of the 33rd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics*. 189-196..

Parameswarappa SS Narayana. 2011. *Target word sense disambiguation system for Kannada language.* , In *Advances in Recent Technologies in Communication and Computing (ARTCom 2011), 3rd International Conference on.* IET. 269-273.

Lesk M. 1986. *Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone.* In *Proceedings of the 5th annual international conference on Systems documentation.* ACM.24-26.

Sinha M Kumar M Pande P Kashyap L Bhattacharyya. 2004. *Hindi word sense disambiguation*, In *International Symposium on Machine Translation, Natural Language Processing and Translation Support Systems*

Gaona Gelbukh Bandyopadhyay. 2009. *Web-based variant of the Lesk approach to word sense disambiguation. In Artificial Intelligence* In *International Symposium on Machine Translation, Natural Language Processing and Translation Support Systems.*.

Kolte S G Bhirud S G. 2009. *WordNet: a knowledge source for word sense disambiguation International Journal of Recent Trends in Engineering.*

Haroon R 2010. *Malayalam word sense disambiguation.* In *Computational Intelligence and Computing Research (ICCIC). IEEE International Conference on. IEEE*. 1-4.

Wallach H M 2004. random fields: An introduction.Technical Reports , In *CIS*. 22.

Zhang J Xu J Zhang Y. 2013. *Name Origin Recognition in Chinese Texts Based on Conditional Random Fields.*, In *2013 I nternational Conference on Information Science and ComputerApplications (ISCA 2013). Atlantis Press*.

McCallum. 2002. *Efficiently inducing features of conditional random fields.*, In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann Publishers Inc*. 403-410.

Lafferty J McCallum A Pereira F C. 2001. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data.*,

Crammer K Singer Y. 2003. . *Ultraconservative online algorithms for multiclass problems.*, *The Journal of Machine Learning Research*. 3:951-991.

Semisupervied Data Driven Word Sense Disambiguation for Resource-poor Languages

Pratibha Rani[†], **Vikram Pudi**[†], **Dipti M. Sharma**[§]
[†]Data Sciences and Analytics Center, [§]Language Technologies Research Center
Kohli Center on Intelligent Systems
International Institute of Information Technology, Hyderabad, India
pratibha_rani@research.iiit.ac.in,{vikram, dipti}@iiit.ac.in

Abstract

In this paper, we present a generic semi-supervised Word Sense Disambiguation (WSD) method. Currently, the existing WSD methods extensively use domain resources and linguistic knowledge. Our proposed method extracts *context based lists* from a small sense-tagged and untagged training data without using domain knowledge. Experiments on Hindi and Marathi Tourism and Health domains show that it gives good performance without using any language specific linguistic information except the sense IDs present in the sense-tagged training set and works well even with small training data by handling the data sparsity issue. Other advantages are that domain expertise is not needed for crafting and selecting features to build the WSD model and it can handle the problem of non availability of matching contexts in sense-tagged training set. It also finds sense IDs of those test words which are not present in sense-tagged training set but their associated sense IDs are present. This feature can help human annotators while preparing sense-tagged corpus for a language by suggesting them probable senses of unknown words. These properties make the method generic and especially suitable for resource-poor languages and it can be used for various languages without requiring a large sense-tagged corpus.

1 Introduction

Word Sense Disambiguation (WSD) is considered as one of the most challenging Natural Language Processing (NLP) task and is described as an AI-complete problem (Navigli, 2009; Mallery, 1988). This is a classification task which involves determining the correct meaning of each word in a sentence/phrase based on the neighboring context words. Humans are very good at judging meaning of words in different contexts but when it comes to automate this task, it becomes very tough. Design of automated WSD methods, both supervised and unsupervised, requires the intuitive knowledge transfer from humans to WSD algorithms via knowledge structures like WordNet (Fellbaum, 1998; Banerjee and Pedersen, 2002), machine readable dictionaries (Lesk, 1986) and sense-tagged training corpus (Navigli, 2009). Creation of such knowledge structures is a costly and time taking process which requires extensive amount of domain resources and linguistic expertise. Along with this, domain expertise is also needed to create and select hand crafted features and rules from the training data which are required in the automated methods. These requirements make it difficult to design a WSD algorithm for (6500+) (Nakov and Ng, 2009) "resource-poor" languages.

The existing literature on WSD methods report that the naive *Most Frequent Sense* (MFS) baseline obtained from a sense-tagged corpus is very hard to beat (Navigli, 2009; Bhingardive et al., 2015b). When (Preiss et al., 2009) tried to refine the selection of most frequent sense by using supplementary linguistic resources like POS tagger and Lemmatizer of the concerned language they found that performance of such a system is limited by the performance of used linguistic resources. This observation shows that for resource-poor languages use of other linguistic resources is not much beneficial in WSD task, since their performances are also dependent on the availability of tagged/knowledge corpus. This inspires us to explore methods for WSD which do not rely on other linguistic resources and can take advantage of contextual information about words and

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 503–512

senses present in the sense-tagged and raw untagged training sets. Also, the challenges of requiring domain expertise and non availability of large sets of sense-tagged data motivated us to develop semi-supervised methods for WSD task. The semi-supervised methods can take advantage of raw untagged data and would require only a moderate or small amount of sense-tagged training data. In semi-supervised scenario, WSD method builds its disambiguation model from a corpus of untagged raw sentences and a set of sense-tagged sentences and is formally defined as:

Using (1) sense IDs set $\Gamma = \{SID_1, SID_2, \ldots, SID_n\}$, (2) **sense-tagged** sentences set $AD = \{St_1, St_2, \ldots St_N\}$, where, $St_i = \langle W_1/SID_i, W_2/SID_j \ldots W_n/SID_k$, W_i is a word and SID_i is a sense ID from Γ and (3) **raw untagged** sentences set $RD = \{RS_1, RS_2 \ldots RS_M\}$, where $RS_i = \langle W_1 W_2 \ldots W_m \rangle$, build a WSD model Θ which outputs the best sense ID sequence $\langle SID_1 SID_2 \ldots SID_l \rangle$ for an input sequence of words $\langle W_1 W_2 \ldots W_l \rangle$.

Here, we propose a semi-supervised WSD method which uses the concept of *context based list* (Rani et al., 2016) to build the WSD model from a set of sense-tagged and raw untagged training corpus. Our proposed method is also influenced by the *one sense per collocation* hypothesis of Yarowsky (1993) which tells that the sense of a word in a document is effectively determined by its *context* (Yarowsky, 1995). Our approach takes help of raw untagged data and expands the notions of context and *context based list* (Rani et al., 2016) to tackle the data sparsity issue. Our method does not require any preprocessing such as, stop/non-content word removal and feature generation and selection from the sense-tagged training corpus. It works without using any additional knowledge structure like dictionary etc., other than the small sense-tagged corpus and moderate sized raw untagged data. This is easily obtainable even for resource-poor languages.

The obtained results show that our method performs well even with very small sized sense-tagged training data for Hindi and Marathi languages and its performance is better than the *Random Baseline* (Navigli, 2009) which selects a random sense for each polysemous test word, comparable to the *Most Frequent Sense* (MFS) baseline that selects the most frequent sense available in the sense-tagged training corpus for each polysemous word and at par with the reported results on the used datasets (Bhingardive et al., 2015a; Bhingardive et al., 2013; Khapra et al., 2011a; Khapra et al., 2011b; Khapra et al., 2008).

Rest of the paper is organized as follows: Section 2 presents related work. Section 3 describes our proposed approach. Section 4 presents and discusses the results and Section 5 concludes the paper and mentions future work directions.

2 Related Work

Generally, all the existing WSD techniques can be categorized into one of the following approaches (Navigli, 2009; Pal and Saha, 2015): i) Knowledge based approach, which uses knowledge structures like, WordNet (Fellbaum, 1998; Banerjee and Pedersen, 2002) or machine readable dictionaries (Lesk, 1986), ii) Supervised approach, which uses machine learning (Kågebäck and Salomonsson, 2016) and statistical methods (Iacobacci et al., 2016) on manually created sense-tagged training corpus. It also requires domain expertise for creating and selecting features and rules to be used for preprocessing and transforming the training data into the form required for designing the algorithm (Navigli, 2009; Iacobacci et al., 2016), iii) Unsupervised approach, which uses large amount of raw untagged training corpus (Pedersen and Bruce, 1997; Lin, 1998) to find word clusters which discriminates the senses of the words in different clusters, or use multilingual parallel corpora (Ide et al., 2002; Bhingardive et al., 2013), a knowledge resource like WordNet (Patwardhan et al., 2007; Chen et al., 2009; Bhingardive et al., 2015b; Bhingardive et al., 2015a) or multilingual dictionary (Khapra et al., 2011a), and iv) Semi-supervised approach, that uses both sense-tagged and untagged data in different proportions with different methods like, co-training with multilingual parallel corpora (Yu et al., 2011), bootstrapping (Yarowsky, 1995; Khapra et al., 2011b), neural network (Taghipour and Ng, 2015; Yuan et al., 2016) and word sense induction (Baskaya and Jurgens, 2016).

All types of WSD algorithms require knowledge structures and resources like, WordNet (Fellbaum, 1998; Banerjee and Pedersen, 2002), machine readable dictionaries (Lesk, 1986), sense-tagged training corpus (Navigli, 2009), parallel corpora and lage untagged raw corpus. Creation

of such knowledge structures and resources is a costly and time taking process which requires extensive amount of domain resources and linguistic expertise. Due to this, for resource-poor languages, special methods are needed which can handle data sparsity issue present in sense-tagged training data and can work with small/moderate set of untagged corpus without requiring knowledge structures and linguistic resources.

To handle the WSD task related challenges of resource-poor languages some specific methods have been proposed. For Chinese language, Yang and Huang (2012) propose handling data sparsity issue by using synonyms for expansion of context, their first method regards synonyms as topic contextual feature to train Bayesian model and second method treats context words made up of synonyms as pseudo training data. Baskaya and Jurgens (2016) propose a Word Sense Induction and Disambiguation (WSID) (Agirre and Soroa, 2007) model in which they combine a small amount of sense-tagged data with information obtained from word sense induction (a fully unsupervised technique that automatically learns the different senses of a word based on how it is used). Yu et al. (2011), Khapra et al. (2011b), Khapra et al. (2011a) and Bhingardive et al. (2013) propose methods to use one language to help other language by means of multilingual parallel corpora, multilingual dictionary, translation and bilingual bootstrapping. Mancini et al. (2016) and Bhingardive et al. (2015a) propose to use word and sense embeddings derived from raw untagged data and WordNet. In this method a large raw corpus is needed to obtain word embeddings.

Bhingardive et al. (2015a), Bhingardive et al. (2013), Khapra et al. (2011a), Khapra et al. (2011b) and Khapra et al. (2008) have reported results on the same dataset which we have used in our experiments. The method used in Khapra et al. (2008) combines sense distributions and sense co-occurrences learned from corpora with semantic relations present in WordNet by specially selecting linguistic features from the sense-tagged data, WordNet, multilingual sense dictionary and a parallel corpus. Khapra et al. (2011b) uses bilingual bootstrapping in which, a model is first trained using the seed annotated data of one language and then it is used to annotate the untagged data of other language and vice versa using parameter projection. Then from both the languages untagged instances annotated with high confidence are added to their seed data and the above process is repeated. Khapra et al. (2011a) uses an unsupervised bilingual Expectation Maximization (EM) based approach requiring synset-aligned bilingual dictionary and in-domain corpora of the concerned language pairs to estimate sense distributions of words in one language based on the raw counts of the words in the aligned synset in the other language. Bhingardive et al. (2013) add use of context in this EM method (Khapra et al., 2011a) and approximate the co-occurrence counts using WordNet-based similarity measures. Bhingardive et al. (2015a) further extends this EM method by using distributional similarity obtained from Word Embeddings to approximate the co-occurrence counts.

3 Proposed Semi-supervised Word Sense Disambiguation Method

Since a context can occur in multiple places in the text, we utilize the contextual similarity property based on *one sense per collocation* hypothesis of Yarowsky (1993) to develop our semi-supervised WSD method. We build upon the concept of *context based list* (CBL) proposed by Rani et al. (2016) for POS-tagging. They call the list of words occurring in a particular context as CBL and use association rule mining (Agrawal et al., 1993) for obtaining effective context based POS tagging rules from the set of tagged and raw untagged training data. We extend their idea by supplementing CBL with the concepts of *extended context list*, *context based sense list* and *context based word list* (defined below) to handle the peculiar problems of WSD due to data sparsity like:

1. Non availability of matching contexts of a word in sense-tagged training set. Use of raw untagged data with concept of *extended context list* helps in dealing with this problem.

2. Non availability of words in sense-tagged training set. Use of raw untagged data with concept of *context based lists* helps in dealing with this problem.

3. Large imbalance in frequencies of senses associated with a word in training set. Defined threshold parameters and *context based lists* help in handling this problem.

Our notion of *context* is a word pair, we use the left and right immediate neighboring words of a

Algo_Present($SIDListSet$, $MWordTaggedListSet$, $MWordUntaggedListSet$, W_t, W_{tl}, W_{tr})

1. **If** test word W_t and its context (W_{tl}, W_{tr}) is present as trigram (W_{tl}, W_t, W_{tr}) in sense-tagged text collection **Then:**
2. Find the corresponding sense IDs of W_t from set $SIDListSet$ and **Return** the sense ID having highest W_t count
3. **Else:**
4. Find set $ExpandTestConPList$ of contexts similar to (W_{tl}, W_{tr}) by finding its Extended Context List using set $MWordTaggedListSet$
5. Find set $ProbTestSIDList$ of all available sense IDs of W_t with their counts from sense-tagged text collection
6. From set $ExpandTestConPList$ find the contexts which are present in sense-tagged text collection with W_t as trigram using set $MWordTaggedListSet$ and from these trigrams select those having highest **ExtContextCount** value in set $ExpandTestConPList$ to make set $maxProbConSet$
7. **For each** context (W_{ptl}, W_{ptr}) of set $maxProbConSet$ **:**
8. Find the sense IDs associated with (W_{ptl}, W_{ptr}) using the set $SIDListSet$ and filter out those which exist in $ProbTestSIDList$ to make set $FinalTestSIDList$
9. **If** $FinalTestSIDList$ is not empty **Then:**
10. **Return** the sense ID from $FinalTestSIDList$ having highest W_t count
11. **Else:**
12. **If** Context Based Word List of context (W_{tl}, W_{tr}) obtained from set $MWordUntaggedListSet$ contains test word W_t **Then:**
13. Find the sense IDs associated with (W_{tl}, W_{tr}) using set $SIDListSet$ and filter out those which exist in $ProbTestSIDList$ to make set $ConFinalTestSIDList$
14. **If** $ConFinalTestSIDList$ is not empty **Then:**
15. **Return** the sense ID from $ConFinalTestSIDList$ having highest W_t count
16. **Else:**
17. **Return** the sense ID from $ProbTestSIDList$ having highest W_t count
18. **Else:**
19. **Return** the sense ID from $ProbTestSIDList$ having highest W_t count

Algo 1: Algorithm to find Sense ID of words present in sense-tagged text collection.

word/sense ID in a sentence/phrase as its context. Formally, in a given trigram $(W_{i-1}\ W_i\ W_{i+1})$ of words, (W_{i-1}, W_{i+1}) word pair is called *context* of W_i. The preceding word W_{i-1} is called *left context* and succeeding word W_{i+1} is called *right context*. Note that, in a text collection there can be multiple contexts available for a word. We use these terms in defining following concepts used in our WSD method:

Single Sense Word List is a list of word instances (with associated single sense ID) which have only one sense ID associated with them in the sense-tagged text collection.

Context Based Word List is a list of word instances from a text collection sharing the same context. For a given context, (W_l, W_r), its *context based word list* is the list of all words W_m having (W_l, W_r) as one of their contexts in the text collection. This list allows to store multiple instances of a word.

Context Based Sense List is a list of sense ID instances from a sense-tagged text collection sharing the same context. For a given context, (W_l, W_r), its *context based sense list* is the list of sense IDs SID_m having (W_l, W_r) as one of their contexts in the sense-tagged text collection. This list can store multiple instances of a sense ID.

Extended Context List: For a given context, (W_l, W_r) of a word W_m, let $PreListSet$ be the set of words obtained from those context based word lists which have left context W_l in their word list and let, $PostListSet$ be the set of words obtained from those context based word lists which have right context W_r in their word list.

Algo_Absent($SIDListSet$, $MWordTaggedListSet$, $MWordUntaggedListSet$, W_t, W_{tl}, W_{tr})

1. For test word W_t find Extended Context List set $ExpandTestConTagList$ of contexts similar to its context (W_{tl}, W_{tr}) using set $MWordTaggedListSet$
2. From set $ExpandTestConTagList$ select context (W_{extl}, W_{extr}) with highest **ExtContextCount** value
3. Find Context Based Word List $TrainExConListTest$ of (W_{extl}, W_{extr}) from $MWordTaggedListSet$
4. **If** $ListSupport(TrainExConListTest) \geq Minsizethreshold$ **Then:**
5. Using $SIDListSet$ find set $ProbTagSenset$ of sense IDs associated with $TrainExConListTest$ having $UniqueSenseSupport \geq (ListSupport(TrainExConListTest) \times Percentagethreshold)$
6. From set $ProbTagSenset$ find and **Return** $Predsentest$ having highest value of $TotalSenseSupport$ and set $Found = True$
7. **If** $Found \neq True$ **Then:**
8. Find Context Based Word List $RawConListTest$ associated with (W_{tl}, W_{tr}) from $MWordUntaggedListSet$ in which W_t is present
9. Find Context Based Word List $TrainConListTest$ of (W_{tl}, W_{tr}) from $MWordTaggedListSet$
10. **If** $ListSupport(RawConListTest) \geq Minsizethreshold$ and $ListSupport(TrainConListTest) \geq Minsizethreshold$ and Number of matching words between $RawConListTest$ and $TrainConListTest$ $\geq$ (size of smaller list among two -1) **Then:**
11. Using $SIDListSet$ find set $ProbTrSenset$ of sense IDs associated with $TrainConListTest$ having $UniqueSenseSupport \geq (ListSupport(TrainConListTest) \times Percentagethreshold)$
12. From set $ProbTrSenset$ find and **Return** $Predsentest$ having highest value of $TotalSenseSupport$ and set $Found = True$
13. **If** $Found \neq True$ **Then:**
14. Find Extended Context List set $ExpandTestConUntagList$ of contexts similar to context (W_{tl}, W_{tr}) using set $MWordUntaggedListSet$
15. From set $ExpandTestConUntagList$ select context (W_{exutl}, W_{exutr}) with highest **ExtContextCount** value
16. Find Context Based Word List $TrainUtExConListTest$ of (W_{exutl}, W_{exutr}) from $MWordTaggedListSet$
17. **If** $ListSupport(TrainUtExConListTest) \geq Minsizethreshold$ **Then:**
18. Using $SIDListSet$ find set $ProbUtSenset$ of sense IDs associated with $TrainUtExConListTest$ having $(UniqueSenseSupport \geq (ListSupport(TrainUtExConListTest) \times Percentagethreshold))$
19. From set $ProbUtSenset$ find and **Return** $Predsentest$ having highest value of $TotalSenseSupport$ and set $Found = True$
20. **If** $Found \neq True$ **Then:**
21. **Return** $NOEXISTSEN$

Algo 2: Algorithm to find Sense ID of words NOT present in sense-tagged text collection.

Let, $FullExtendConListSet$ be the set of all contexts (W_{pre}, W_{post}) prepared by taking word W_{pre} from $PreListSet$ and word W_{post} from $PostListSet$. Then, *extended context list* is the list of all those contexts from $FullExtendConListSet$ which have W_m in their context based word list. This list contains contexts similar to the given context (W_l, W_r). There is a count value **ExtContextCount** associated with each context present in *extended context list* which shows how many word combinations from $PreListSet$ and $PostListSet$ generated that context.

For a list of words L, in which multiple instances of a word can be present, we define following parameters:

ListSupport(L) is defined as the number of unique words present in L.

UniqueSenseSupport of a particular sense ID, SID, is defined as the number of unique words of L which have SID associated with them in the sense-tagged text collection.

TotalSenseSupport of a particular sense ID, SID, is defined as the total number of words of L (includes repeated occurrences of a word with a sense ID) which have SID associated with them in the sense-tagged text collection.

Minsizethreshold parameter defines the minimum number of words required to be present in a Context Based Word List to consider it for finding sense of words not present in sense-tagged text collection.

Percentagethreshold parameter is used for calculating percentage of words supporting a particular sense ID in a list of words L.

Overview of our WSD method

In the training phase, using a sliding window of size three, we collect all the *context based word lists*, *context based sense lists*, *single sense word list*, word and sense counts from the sense-tagged and raw untagged text collection in a single iteration, taking care of the sentence boundaries. Then in testing phase, Algo 1 and Algo 2 are used to find sense IDs of test words according to their presence/absence in the sense-tagged training set. Algo 1 always provides an output for test words present in sense-tagged training set but Algo 2 returns $NOEXISTSEN$ when it is not able to find any valid sense ID for test words not present in sense-tagged training set.

Both the algorithms use directly available immediate context information and indirectly available extended context information from the sense-tagged and raw untagged text collection in a priority order to handle the issues of non availability of matching contexts and imbalance in sense frequencies associated with a word in sense-tagged training set. Information obtained from sense-tagged set is given higher priority. Algo 2 uses raw untagged set to handle issue of non availability of words in sense-tagged training set and takes help of the defined support and threshold parameters to make confident choice of sense ID. Due to these properties it is able to find sense IDs of those test words also which are not present in sense-tagged training set but their associated sense IDs are present. The detailed steps involved in our WSD method are given in Section 3.1.

3.1 Word Sense Disambiguation Method

Following steps are used in our WSD method:

1. Find *Single Sense Word List* from the sense-tagged text collection.
2. Find set **SIDListSet** of *Context Based Sense Lists* of sense IDs from sense-tagged text collection.
3. Find set **MWordTaggedListSet** of *Context Based Word Lists* of words from sense-tagged text collection.
4. Find set **MWordUntaggedListSet** of *Context Based Word Lists* of words from raw untagged text collection.
5. If test word, W_t, present in sense-tagged text collection and is also present in *Single Sense Word List* then output associated sense ID. Else, find its context (W_{tl}, W_{tr}) from test sentence and apply Algo 1.
6. If test word, W_t, is not present in sense-tagged text collection then find its context (W_{tl}, W_{tr}) from test sentence and apply Algo 2.

4 Results and Discussion

We have used publicly available Health and Tourism domain sense-tagged corpus of Hindi and Marathi languages created by IIT Mumbai[1] (Khapra et al., 2010) and Hindi language raw untagged Health and Tourism domain ILCI data (Jha, 2010). Table 2 gives the dataset details. Table 1 shows average 4-fold cross validation results obtained by our algorithm for polysemous test words which are not present in the sense-tagged training set. Table 3 presents the average 4-fold cross validation results obtained for polysemous test words along with *Random Baseline* and MFS baseline results.

[1]Available at `http://www.cfilt.iitb.ac.in/wsd/annotated_corpus/`

The results are presented in terms of Precision, Recall and F-Score accuracy measures as defined below (Navigli, 2009):

$$Precision = \frac{\text{No. of correctly predicted test words}}{\text{Total No. of predicted test words}} \quad (1)$$

Here, Total No. of predicted test words = (Total No. of test words - Test words flagged $NOEXISTSEN$ by algorithm).

$$Recall = \frac{\text{No. of correctly predicted test words}}{\text{Total No. of test words}} \quad (2)$$

$$FScore = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

Table 1: Average 4-fold cross validation results obtained by our algorithm for polysemous test words NOT present in the sense-tagged training corpus.

Dataset	Precision (%)	Recall (%)	FScore (%)
Hindi Tourism	28.93	22.90	25.56
Marathi Tourism	34.50	12.0	18.0
Hindi Health	31.65	25.41	28.19
Marathi Health	32.43	8.72	13.74

The results of Table 1 shows the advantage of our approach in terms of ability to find sense IDs of those test words also which are not present in the sense-tagged training set but their associated sense IDs are present. To the best of our knowledge, currently supervised and semi-supervised WSD methods do not handle words absent in the sense-tagged training corpus. The *Random Baseline* and MFS baseline methods also can't find sense IDs for words which are absent in the sense-tagged training set. This ability can be used as a tool to help human annotators by suggesting them probable senses of unknown words while preparing sense-tagged corpus for a language.

To study the effect of parameter values on our approach, we experimented with parameter values $Minsizethreshold = 3, 5, 10$ and $Percentagethreshold = 0.5, 0.8$ and observed that variation in obtained results is very less ($\pm 0.5\%$) which shows that our approach is not very sensitive towards parameter values in this range of values. Following parameter values generated best results for our approach presented in Tables 1, 3 and 5: 1) For Hindi Tourism, $Minsizethreshold = 5$ and $Percentagethreshold = 0.8$. 2) For Hindi Health, $Minsizethreshold = 3$ and $Percentagethreshold = 0.8$. 3) For Marathi Tourism, $Minsizethreshold = 3$ and $Percentagethreshold = 0.5$. 4) For Marathi Health, $Minsizethreshold = 3$ and $Percentagethreshold = 0.5$. Our approach uses both the sense-tagged and raw untagged datasets of each domain mentioned in Table 2. We have divided the original Marathi Health and Tourism datasets into two exclusive parts and used one part as raw untagged set and other as tagged set.

Table 3 shows that results of our approach are better than the *Random Baseline* results and very close to the MFS baseline results. We can't directly compare our results with the earlier reported results (see Table 4) on these dataset by Bhingardive et al. (2015a), Bhingardive et al. (2013), Khapra et al. (2011a), Khapra et al. (2011b) and Khapra et al. (2008) due to difference in dataset size and content.

By observing the difference between reported accuracies of approach used by Khapra et al. (2008) and the MFS baseline results reported by them we can conclude that our simple generic approach gives results close to MFS baseline without using any complex feature selection process (domain based and generic) and without requiring too many linguistic and domain resources. For Hindi Tourism, Marathi Tourism and Hindi Health domains our results are better than the results reported by Bhingardive et al. (2015a), Bhingardive et al. (2013), Khapra et al. (2011b) and Khapra et al. (2011a) without using huge raw untagged data and without using any linguistic and domain resources like WordNet, a large multilingual parallel corpus or a multilingual dictionary which are required by the other methods.

Table 5 presents results for experiments with sense-tagged set size smaller than 100×10^3 words and shows that for small training set sizes (less than 50×10^3 words), Recall of our algorithm is better than MFS and Precision and F-Scores are in close range. Hence, it is a good choice for resource-poor languages, especially for those languages for which resources are in development phase. These results and our other experiments show that as sense-tagged training data size increases performance of our method also improves.

Table 2: Statistics of sense tagged and raw untagged datasets.

Dataset	Total No. of Sentences	Total No. of Words	No. of unique Words	No. of unique Sense IDs	Total No. of Polysemous Words	No. of unique Polysemous Words
Hindi Tourism sense-tagged	15395	424836	33500	8088	243959	5015
Marathi Tourism sense-tagged	13914	305337	54780	6307	141019	6758
Hindi Health sense-tagged	8001	189677	13356	4405	108006	2321
Marathi Health sense-tagged	6344	119764	21720	3643	47451	2790
Hindi Tourism raw untagged	24999	424128	29368	-	-	-
Hindi Health raw untagged	24461	447330	21811	-	-	-
Marathi Tourism raw untagged	2011	35208	11104	-	-	-
Marathi Health raw untagged	577	13468	4156	-	-	-

Table 3: Average 4-fold cross validation results obtained for polysemous test words.

Dataset	Our Approach			Random Baseline			MFS		
	Precision (%)	Recall (%)	FScore (%)	Precision (%)	Recall (%)	FScore (%)	Precision (%)	Recall (%)	FScore (%)
Hindi Tourism	76.22	76.14	76.18	39.39	39.39	39.39	78.66	78.27	78.46
Marathi Tourism	64.80	64.03	64.41	45.61	45.61	45.61	66.0	64.80	65.39
Hindi Health	69.97	69.79	69.88	45.47	45.47	45.47	71.45	70.72	71.08
Marathi Health	60.11	59.12	59.61	48.01	48.01	48.01	60.93	59.58	60.24

Table 4: Average 4-fold cross validation F-Score (%) results obtained for polysemous test words of various datasets by our approach and other WSD algorithms.

Algorithms	Hindi Tourism	Marathi Tourism	Hindi Health	Marathi Health
Our Approach	**76.18**	**64.41**	**69.88**	59.61
Bhingardive et al. (2015a)	-	-	60.94	61.30
Bhingardive et al. (2013)	60.70	58.67	59.63	59.77
Khapra et al. (2011a)	53.87	55.20	54.64	58.72
Khapra et al. (2011b)	60.67	61.90	57.99	**64.97**
Khapra et al. (2008)	**74.10**	**74.40**	**74.20**	**78.70**

To study the effect of raw untagged data size, for a particular size sense-tagged training set we varied the raw untagged data size in the range of 2×10^3 to maximum possible for that dataset and observed that as raw untagged data size increases the number of correctly predicted test words not existing in sense-tagged training set also increases which adds to the overall performance of our approach.

5 Conclusions and Future Work

In this paper, we proposed a generic semi-supervised method for Word Sense Disambiguation (WSD) task which uses concept of context based lists and extended context lists. It makes the WSD model without using domain knowledge from a small set of sense-tagged corpus along with raw untagged text data as training data. It works well with small training data also and handles data sparsity issue. It does not require domain expertise for crafting and selecting features to be used in the algorithm and outputs senses of those test words also which are not present in sense-tagged training set but their associated senses are present. It is generic enough to be used for WSD task of various languages without requiring a large sense-tagged corpus and is especially suitable for resource-poor languages. Our exper-

Table 5: Results obtained for polysemous test words for various sense-tagged training set sizes ($\leq 100 \times 10^3$ words).

Dataset	No. of Polysemous Test words	Sense tagged set size	Our Approach				MFS		
			Untagged set size	Precision (%)	Recall (%)	FScore (%)	Precision (%)	Recall (%)	FScore (%)
Hindi Tourism	45721	36457	424128	72.33	**70.40**	71.35	75.38	69.96	72.57
		38377		73.24	**71.32**	72.27	76.87	71.06	73.85
		76436		74.31	73.50	73.90	76.87	73.81	75.31
Marathi Tourism	33316	21747	35208	62.22	**47.77**	54.04	62.85	47.27	53.96
		43251		62.06	**53.05**	57.20	62.73	52.56	57.20
		85296		63.06	**58.16**	60.51	63.79	57.89	60.70
Hindi Health	21648	16936	447330	51.89	**49.35**	50.59	56.99	47.23	51.65
		31144		52.49	**50.93**	51.7	56.26	50.24	53.08
		59035		59.93	59.11	59.52	63.45	60.18	61.78
Marathi Health	10340	7665	13468	77.96	**57.89**	64.44	78.51	57.88	66.64
		15678		73.21	**61.94**	67.12	73.43	61.52	66.95
		33753		70.44	65.62	67.94	71.48	66.06	68.67
		75379		64.66	63.09	63.87	65.37	63.36	64.35
		94411		64.40	63.44	63.92	65.78	64.45	65.11

iments on Tourism and Health domains of Hindi and Marathi languages show good performance without using any language specific linguistic information.

Future work would be to test it on other languages including English. Further exploration can be done to enhance the property of finding sense IDs of non existing words. We can also try to include more generic features in the algorithm to enhance performance.

References

Eneko Agirre and Aitor Soroa. 2007. Semeval-2007 Task 02: Evaluating Word Sense Induction and Discrimination Systems. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval '07)*, pages 7–12. Association for Computational Linguistics.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining Association Rules Between Sets of Items in Large Databases. In *SIGMOD'93*, pages 207–216.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 136–145. Springer.

Osman Baskaya and David Jurgens. 2016. Semi-supervised Learning with Induced Word Senses for State of the Art Word Sense Disambiguation. *J. Artif. Int. Res.*, 55(1):1025–1058.

Sudha Bhingardive, Samiulla Shaikh, and Pushpak Bhattacharyya. 2013. Neighbors Help: Bilingual Unsupervised WSD Using Context. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL*, volume 2: Short Papers, pages 538–542.

Sudha Bhingardive, Dhirendra Singh, V Rudramurthy, and Pushpak Bhattacharyya. 2015a. Using Word Embeddings for Bilingual Unsupervised WSD. In *Proceedings of the 12th International Conference on Natural Language Processing (ICON 2015)*, pages 59–64.

Sudha Bhingardive, Dhirendra Singh, Rudra Murthy V, Hanumant Harichandra Redkar, and Pushpak Bhattacharyya. 2015b. Unsupervised Most Frequent Sense Detection using Word Embeddings. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1238–1243.

Ping Chen, Wei Ding, Chris Bowes, and David Brown. 2009. A Fully Unsupervised Word Sense Disambiguation Method Using Dependency Knowledge. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for Word Sense Disambiguation: An Evaluation Study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL*, volume 1.

Nancy Ide, Tomaz Erjavec, and Dan Tufis. 2002. Sense Discrimination with Parallel Corpora. In *Proceedings of the ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions - Volume 8*, pages 61–66.

Girish Nath Jha. 2010. The TDIL Program and the Indian Langauge Corpora Intitiative (ILCI). In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. European Language Resources Association.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional LSTM. *CoRR*, abs/1606.03568.

Mitesh Khapra, Pushpak Bhattacharyya, Shashank Chauhan, Soumya Nair, and Aditya Sharma. 2008. Domain specific iterative word sense disambiguation in a multilingual setting. In *Proceedings of International Conference on NLP (ICON 2008)*.

Mitesh M. Khapra, Anup Kulkarni, Saurabh Sohoney, and Pushpak Bhattacharyya. 2010. All Words Domain Adapted WSD: Finding a Middle Ground Between Supervision and Unsupervision. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1532–1541. Association for Computational Linguistics.

Mitesh M Khapra, Salil Joshi, and Pushpak Bhattacharyya. 2011a. It Takes Two to Tango: A Bilingual Unsupervised Approach for Estimating Sense Distributions using Expectation Maximization. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 695–704. Asian Federation of Natural Language Processing.

Mitesh M. Khapra, Salil Joshi, Arindam Chatterjee, and Pushpak Bhattacharyya. 2011b. Together We Can: Bilingual Bootstrapping for WSD. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 561–569. Association for Computational Linguistics.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 768–774.

John C Mallery. 1988. Thinking about foreign policy: Finding an appropriate role for artificially intelligent computers. In *Master's thesis, MIT Political Science Department*.

Massimiliano Mancini, José Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. 2016. Embedding Words and Senses Together via Joint Knowledge-Enhanced Training. *CoRR*, abs/1612.02703.

Preslav Nakov and Hwee Tou Ng. 2009. Improved Statistical Machine Translation for Resource-poor Languages Using Related Resource-rich Languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3*, pages 1358–1367. Association for Computational Linguistics.

Roberto Navigli. 2009. Word Sense Disambiguation: A Survey. *ACM Computing Survey*, 41(2):10:1–10:69.

Alok Ranjan Pal and Diganta Saha. 2015. Word sense disambiguation: A Survey. *CoRR*, abs/1508.01346.

Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2007. UMND1: Unsupervised Word Sense Disambiguation Using Contextual Semantic Relatedness. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 390–393.

Ted Pedersen and Rebecca Bruce. 1997. Distinguishing Word Senses in Untagged Text. In *eprint arXiv: cmp-lg/9706008*.

Judita Preiss, Jon Dehdari, Josh King, and Dennis Mehay. 2009. Refining the Most Frequent Sense Baseline. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, DEW '09, pages 10–18. Association for Computational Linguistics.

Pratibha Rani, Vikram Pudi, and Dipti Misra Sharma. 2016. A semi-supervised associative classification method for POS tagging. *International Journal of Data Science and Analytics*, 1(2):123–136.

Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-Supervised Word Sense Disambiguation Using Word Embeddings in General and Specific Domains. In *HLT-NAACL*, pages 314–323.

Zhizhuo Yang and Heyan Huang. 2012. Chinese word sense disambiguation based on context expansion. In *COLING (Posters)*, pages 1401–1408.

David Yarowsky. 1993. One sense per collocation. In *Proceedings of the workshop on Human Language Technology*, pages 266–271.

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *ACL*, pages 189–196.

Mo Yu, Shu Wang, Conghui Zhu, and Tiejun Zhao. 2011. Semi-supervised learning for word sense disambiguation using parallel corpora. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 3, pages 1490–1494. IEEE.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. *arXiv preprint arXiv:1603.07012*.

Notion of Semantics in Computer Science
A Systematic Literature Review

Gollapudi VRJ Sai Prasad
Software Engineering Research Center
IIIT-Hyderabad
Gachibowli, Telangana, India
saigollapudi1@gmail.com

Venkatesh Choppella
Software Engineering Research Center
IIIT-Hyderabad
Gachibowli, Telangana, India
venkatesh.choppella@iiit.ac.in

Abstract

In this paper we report on a Systematic Literature Review where we explored the notion of semantics in Computer Science (CSE) literature. Our goal was 1) to surface how the idea of semantics has been used and represented, and 2) to surface its publication pattern in CSE. Our automated search in 5 CSE repositories yielded 653 relevant papers, emerging from multiple disciplines and geographies, spanning a period from year 1967 to 2017. We shortlisted 50 representative samples to study. This literature review was motivated by an external Web Accessibility effort in which we wanted to understand how to influence the various meanings that a variety of human end-user could derive by varying the computer rendering of a given content. The results of the SLR indicate that 44% of papers do have their own definition, almost all are formal in their presentation, and 94% of them have a notion of semantics that favors the computer as a processor. We observe the limited human oriented focus on semantics in CSE, and suggest such semantics focus as an area of potential study.

1 Introduction

In the scenario of a human interacting with a computer, meaning is getting produced and processed. We are interested in impacting the notion of this *meaning* that is getting created in the human.

Philosophers and Linguists have routinely used the term "Semantics" to represent the notion of meaning. Now, this notion of semantics has been carried over to Computer Sciences where it has been applied in Natural Language Processing, Programming Languages, Web, Software Engineering etc.

Motivated by impacting human meaning and human sense-making, we are therefore transitively interested in the notion of semantics. However, even for a contained area like Computer Science (CSE), this term has not been unambiguously defined for similar and consistent use universally. Many disciplines within CSE have all made use of this notion of Semantics, but in their own way.

1.1 Goal of this Paper

The goal of this paper is to conduct a Systematic Literature Review on the notion of semantics in CSE. In particular we aim to study the use of this term in the CSE, at least in such disciplines as Programming Languages (PL), Software Engineering (SE), Compilers, Web and NLP. The overall goal being, 1) to surface how the term of semantics has been used and represented in these said disciplines, and 2) to surface the publication pattern on this topic.

1.2 Background, Context & Motivation

In human computer interaction, there is obviously 1) a human, 2) a computing system and 3) an engagement or interaction between the two. The engagement could either be passive (as in browsing or viewing), or active, as in querying or selecting something on the system. In such scenarios, humans are said to be deriving meaning from the representation presented by the computer. The modality for representation can be text, image, audio, video etc. More interactive representation(al experiences) can be animation, video, user interfaces etc. In the case of interaction (as in inputting or programming by the human), the computing system is also processing data to derive meaning. Apparently, both the human and the system can be seen as two processing agents.

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 513–522

The notion of meaning and semantics can, therefore, be applied to either of the two agents. Our interest, however, is on the human formulating meaning. From an information delivery point-of-view, the idea of how meaning is extracted, constructed or possessed by the human is studied by Psychologists, Cognitive Scientists and Information Processing researchers. On this side, topics like Sense-making (Russell et al., 1993), User Experience, Semantic Interaction (Endert et al., 2012) etc. emerge.

As a compliment to the human sense making experience, on the computing side, we may also look at how something can be constructed to deliver a particular meaning. Web Accessibility researchers, claim that currently web content is primarily designed for a majority in mind (Prasad et al., 2014). And that it may not suffice for the individualized needs of a minority of users (Prasad, 2017).

A color blind person, for example, may not benefit in the same way as a non-body disabled user. So, in this regard, on the computing system side of the human computer interaction, does there exist a platform that would enable the creation and simultaneous co-existence of multiple representations for the varying needs of a diverse human end users? Is there sufficient motivation for a system that can *renarrate* and simultaneously have multiple representations of some source text (Prasad, 2017)? That is, a system equally being able to produce colorful content for the majority of users, high contrast and appropriately rendered visuals for the color blind, braille for the visually impaired, in vernacular for the non-English speakers, in tables, diagrams and scientific explanations for the learned etc. These questions form the background context and motivation for our study of semantics in CSE.

1.3 Semantics as "Meaning"

Online dictionary[1] describes Semantics as "the meaning, or an interpretation of the meaning, of a word, sign, sentence, etc." From a linguistic point of view, it relates Semantics to "the study of meaning". Webster's dictionary[2] too shares a similar explanation, and calls semantics as "the study of meanings".

From a human computer interaction point view, the study of semantics can be related to the study of meaning for either the human or the computer. We are keen to uncover how semantics research in CSE has defined and explored this topic.

1.4 SLR - A Research Tool

As already stated, our larger goal is to understand how best to represent either information or data on the system so that it may create the right meaning to the human. To that end we wanted to conduct an exploratory Literature Review for such a social applicable, human oriented web application space.

SLRs have been popularized as a Evidence Based Software Engineering (EBSE) research tool by Kitchenham et al. in a seminal paper (Kitchenham et al., 2004) presented at ICSE 2004, which is a prominent conference for Software Engineers. In particular SLRs have been suggested as a systematic way of exploring a problem space and thus have been suggested as valuable first step in a PhD research effort (Kitchenham et al., 2004).

While SLRs have been popular in the fields of medical sciences, their use in CSE has been limited. However, we are now beginning to find SLRs in various areas of CSE. SLRs are now being published in Information Systems (Okoli and Schabram, 2010), Software Engineering (Kitchenham et al., 2009), Programming Languages (Major et al., 2012), Web (Doğan et al., 2014), Model Driven Engineering (Santiago et al., 2012) etc.

2 Research Method

This SLR follows the guidelines given in (Keele and others, 2007) and is also informed by DARE[3] criteria for SLR.

2.1 Research Questions

The research questions put forth for the documents surfaced by our search strategy (given in section 2.2) include:

RQ1: Was there a definition for semantics in the paper?

RQ2: Was the notion of semantics general, or did it have some sub categories? What where they?

RQ3: Is the notion of semantics oriented towards the human or the computer?

RQ4: What sort of precision did it have in its definitions?

[1] http://www.dictionary.com/browse/semantics?s=t

[2] https://www.merriam-webster.com/dictionary/semantics

[3] Database of Abstracts of Reviews of Effects (DARE): https://www.ncbi.nlm.nih.gov/pubmedhealth/about/DARE/

RQ5: Which research domain did the paper represent within CSE?

RQ6: When was the research published?

2.2 Search Strategy

For the SLR we conducted an automated search, which included five of the most commonly used CSE bibliography repositories. See Table 1. Each of the databases were searched, in the stated order of priority, on the following aspects: 1) the queried records must be CSE papers, 2) they must have the word "semantic" in their title, and 3) they must have the term "definition of semantics" in their body of text. Table 1 lists the exact string and the restrictions that were used for our automated search.

2.3 Paper Selection

Paper selection was based on a set of inclusion, exclusion and quality criteria. The **inclusion criteria** required the document to fulfill the search string, be a peer reviewed primary study, and be an accessible document on the web. Papers with zero citations, papers that were essentially Patents, papers that were on non-CSE topics (like biology/genome) were **excluded**. **Quality criteria** consisted of only selecting papers that were considered long publications (i.e. had to be more than 4 pages), had to be peer reviewed, and had to have some citations.

2.3.1 Selection Process

Once a paper fulfilled our inclusion, exclusion and quality criteria, it was entered into our initial corpus for individual selection. The initial corpus was maintained as Bibtex files in Microsoft Excel worksheet. We expected our initial corpus to be quite large, we planned on manually shortlisting it into a handlable size for evaluation. This shortlisting process was done on Excel by two outside judges. Our aim was to reduce the initial corpus into a more practical size of 50 representative samples. These set of shortlisted records were then to be fed into a document manager to surface the full length documents from the web. For PDF management we used Qiqqa tool[9], and for Bibtex management, we used JabRef tool. This set of 50 shortlisted records, complete with their full body content were then positioned on the Qiqqa tool for data collection.

[9]Desktop v.79s for Windows; source:`http://www.qiqqa.com/`

The data that was used for filtering was Title, Keywords, publication meta-data (like the publisher, journal name, issue details etc), and in some cases Abstracts as well.

2.4 Data Extraction

The intent of this phase is to ensure that we collect appropriate data from each earmarked paper to answer our earlier stated SLR research questions. Here is the criteria that was used for each questions:

RQ1: We used the document management tools[10] to search for various definitions found in the papers. If there were any definitions on the topic related to semantics then we took it as a *YES* count. Else, it was counted as a *NO*.

RQ2: We searched the surfaced papers to uncover the various contexts[11] in which the word semantic was used. If there were any repeatedly used *sub-concepts* of semantics then we recorded it. At the end we expected to have a bag of semantics related concepts and ideas that would form the base for where the CSE research was headed.

RQ3: One key differentiation we wanted to make was to whom the semantics was being made relevant to. Was it the *human* (as a processor of rendered information), or was it the *computer*[12] (as a processor of the input information)? We scanned papers to see how the definitions of semantics were oriented, and incremented the relevant "H" or "C" count as appropriate.

RQ4: Through this question we wanted to see if the papers presumed an earlier (elsewhere) defined notion, or if they took the trouble to define their own working definition. In some cases we expected to also have some loosely defined terminology. So, our measure was on the precision: Was the definition *formal* (with logic and mathematics)? Or, was it *informal* - as in just by English text? Or (as in RQ1) was there *no* definition at all? This was checked and recorded.

RQ5,6: For the last two questions we collected meta data on the publications. Here we wanted to see where the research was emerging from. We wanted to understand which *domains* were active in this research and the *year* of publication.

In addition to the above highlighted data, we

[10]which, in our case, was Qiqqa Desktop

[11]The word context is refers to the research narrative and not the context of corpus within some research.

[12]We treated these two as mutually exclusive even though they need not be.

Databases	Search String	Restrictions	Hits
ACM Digital Library[4]	acmdlTitle:(+semantics) AND content.ftsec:(+"definition of semantics")		282
IEEE Xplore[5]	((definition of semantics) AND "Publication Title":semantics)		67
Science Direct[6]	TITLE(semantics) and (definition of semantics) AND LIMIT-TO(topics, "theoretical computer,logic program,program,definition").	Advanced Search/Expert Search tag; used no theoretical, no books filters	342
SpringerLink[7]	"definition of semantics"	"definition of semantics" anywhere and "semantics" in title; used Articles, Computer Science, English filters	61
Wiley Inter Science[8]	definition of semantics in All Fields AND semantics in Publication Titles	Advanced Search	33
		total	790

Table 1: The CSE bibliographic repositories that were used in the automated search.

also collected such publication related meta data as: Title, Keywords, Author names, Publication, Year, and in some cases, even the Abstract. We used Bibtex for the extraction of this information from the online bibliographies.

Essentially, through this data collection, we sought to surface how computer science research viewed semantics with respect to their own work, and to see how these ideas tallied with our idea of influencing meaning in the minds of an end user.

3 Results

Our initial automated search extracted 790 records, of which 5 records were malformed and irretrievable. In this initial corpus we were able to identify 21 repeat records, 87 with no "semantics", and 32 short papers. That is, overall 140 were eliminated from this initial corpus, resulting in 653 retrievable pruned set of records.

In studying the initial corpus we found that our collection was indeed quite diverse: For example, the publication dates ranged from 1967 to 2017. The locations of publications at least included USA, UK, Germany, Australia, South Africa, Netherlands, Switzerland and Canada. The covered disciplines included Theoretical CSE, Knowledge Engineering, Formal Methods, Programming Languages, Logic Programming, Semantics, Web, Linguistics, Systems, Multimedia, Software Engineering, Artificial Intelligence etc. Even Biology/genome related publications were captured.

From this diverse sample set of 653 records, as per our selection process, we then needed to shortlist a smaller sample size of just 50 records. We used two external judges to help us identify 50 representative samples from the original list of 653. While the choice was somewhat arbitrary, it was still ensured that the reduced set too was sufficiently diverse and indicative of the larger set of 653. Tables 2 and 3 provide a listing of these finalized studies.

The earmarked 50 records were converted into a shortlisted bib file by use of the JabRef tool[13]. The bib file was used by our document manager, Qiqqa, to import the full content. The files were imported from online document repositories given by Table 1. Finally, for subsequent steps involving data extraction, the same Qiqqa tool was then used to manage the 50 uploaded PDFs.

Here is a brief summary of what was uncovered through our data collection process:

3.1 RQ1: Definition

In the first RQ1 we wanted to understand how many, if any, actually even bothered to define the notion of semantics in their research. Our initial presumption was that while the idea of semantics and usage of the term was rampant, the definition was most likely ambiguous and perhaps not sufficiently formal.

The results of our SLR contradicted our assumptions. The data informed us that while 56% (that is, 28 out of 50) papers were indeed assuming a pre-existing notion and definition of semantics, there were also the other 44% that indeed contained definitions. That is, 22 of the 50 samples actually had expressed their notion of semantics.

Upon investigation we found that most of them were either having special applications or were defining niche terms related to semantics. For example, S5[14] for these references. defined the notion of "meaningfulness", S6 had Context Free Grammar (CFG) related semantics, S16 had defi-

[13]Our JabRef tool was part of our TexStudio Latex document editor, and was supported by Qiqqa.

[14]See Tables 2, 3 for listing of 50 sample studies we used in our SLR. Due to the long length, it has been divided into two parts.

ID	Author	Formal Definition Present	hum/com	subtypes	domains
S1	(Ray et al., 1998)	none	comp	correctness	database
S2	(Haghverdi and Scott, 2005)	none	comp	denotational	prog lang
S3	(Hans Bruun, 1991)	none	comp	static	prog lang
S4	(Alexandre Rademaker, 2005)	none	comp	logic	software eng
S5	(Lavelli et al., 1992)	yes - meaningfulness	comp	multilevel	systems
S6	(Vykhovanets, 2008)	yes - CFG related	comp	general	prog lang
S7	(Schwarcz, 1969)	yes	comp	general	nlp
S8	(Juba and Sudan, 2008)	none	comp	universal	nlp
S9	(Winsborough, 1992)	none	comp	graph	compiler
S10	(Andrew Butterfield, 2006)	none	comp	general	compiler
S11	(Glesner, 2005)	none	comp	general	compiler
S12	(Dan R Ghica, 2012)	none	comp	game, denotational	game
S13	(Matos et al., 2010)	none	hum	general	web services
S14	(Pittarello and De Faveri, 2006)	none	comp	general	web
S15	(Yong et al., 2004)	none	hum	general	urban planning
S16	(Alexey L. Lastovetsky, 2001)	yes - abstract lang	comp	general	prog lang
S17	(Benveniste et al., 1991)	none	comp	general	prog lang
S18	(Perdrix, 2008)	yes - for a quantum program	comp	quantum	theoretical, logic
S19	(Bochman, 1998)	yes- for logic programs	comp	stationary and stable class	theoretical, logic
S20	(Blair, 1982)	none	comp	general	theoretical, logic
S21	(Cox and Dang, 2010)	none	comp	general	prog lang
S22	(Velbitskiy, 1977)	yes - a meta lang	comp		prog lang
S23	(Menezes, 2008)	yes - for aspect oriented	comp	denotational, operational, action	prog lang
S24	(Zhou and Zhang, 2017)	none	comp	stable model	theoretical, logic
S25	(Toch et al., 2007)	yes - narrow web service semantics	comp		web services

Table 2: Listing of 50 sample studies we used in our SLR. Due to the long length, it has been divided into two parts. This represents the first part.

nitions to be used in an abstract language, S23 applied semantics to Aspect Oriented Programming concepts etc. Again, the existence of such definitions confirmed to us that work with semantics is not as arbitrary as we had initially presumed.

3.2 RQ2: Subcategories

The intent here was to understand how generic was the application of semantics. Our results suggest that there are indeed many research works and disciplines that discuss semantics at a very high level, but there are also those that sufficiently focused in on the sub topics within semantics.

In our collection, the subtopics that were explored included: denotational semantic (S2), static semantic (S3), logic semantics (S4), multilevel semantics (S5), universal semantics (S8), graph semantics (S9), game semantics (S46), quantum semantic (S18), stationary semantic (S19), stable class (S19), operational semantics (S23), action semantics, stable model semantics (S24), trace semantics (S37) etc. Other notion of semantics include: semantic correctness (S1), semantic relatedness (S27), semantic distance (S27), semantic forgetting, semantic compatibility (S42,44), timed semantics, semantic spaces, semantic models, semantic similarity etc.

3.3 RQ3: Human Vs. Computer Semantics

Through this RQ3 we wanted to uncover a presumption that most of the notion of semantics in CSE was computer oriented and not human oriented. The SLR results confirmed this. We found that 47 out of 50 papers were indeed meant for computers as the processing agent. Only 3 out of the 50 were designed for human as the processing

ID	Author	Formal Definition Present	hum/com	subtypes	domains
S26	(Kravicik and Gasevic, 2006)	none	comp	general	web services
S27	(Xu et al., 2006)	yes - for relatedness of keywords	comp	relatedness, distance	ontology
S28	(Emmon Bach, 2008)	none	comp	general	linguistics
S29	(Bergmann and Gil, 2014)	none	comp	general	workflows
S30	(Dasiopoulou et al., 2010)	none	comp	general	image analysis
S31	(Biancalana et al., 2013)	none	comp	general	social web
S32	(Paolini, 2009)	none	comp	general	theoretical, logic
S33	(Boute, 1988)	yes -for SDL	comp	denotational	systems
S34	(Papaspyrou, 2001)	yes - for C	comp	denotational	prog lang
S35	(Lobo et al., 1991)	yes - Logic	comp		logic
S36	(Broy and Lengauer, 1991)	yes - Logic	comp	predicative, denotational	theoretical, logic
S37	(Puntigam, 1997)	none	comp	trace	prog lang
S38	(Jasmin Christian Blanchette, 2008)	yes - alternatives presented	comp	operational	prog lang
S39	(Thomas Eiter, 2008)	yes - for answer sets	comp	forgetting, stable model	theoretical, logic
S40	(Ouksel and Sheth, 1999)	none	comp	general	Global Info Systems (GIS)
S41	(Millard et al., 2005)	yes- for hypertext	comp	general	hypertext, logic
S42	(Zeng et al., 2006)	yes - compatibility	comp	compatibility	prog lang
S43	(Wehrman et al., 2008)	yes - for ORC	comp	operational, denotational, timed	theoretical; logic
S44	(Zeng et al., 2005)	yes - compatibility	comp	compatibility	web services
S45	(Benthem, 2005)	none	comp	general	logic
S46	(Kessing et al., 2012)	none	hum	general	game
S47	(Baroni and Lenci, 2010)	none	comp	spaces, models, similarity	distributed memory; database
S48	(Abiteboul and Hull, 1987)	yes - IFO database model	comp	general	database
S49	(da Silva et al., 2012)	none	comp	general	workflows; web services
S50	(Titov and Klementiev, 2011)	yes- bayesian parsing	comp	general	nlp

Table 3: Part two, or the remaining listing of 50 sample studies we used in our SLR.

agent.

Upon further investigation, these 3 were either using a specialized concept of semantics or were geared towards a social application. For example S15 had to use human understandable terms like Roof, Window, Gate, Shell, Wall etc to link the graphics to urban planning. S13 used a cell component ontology, and S46 focused on real world physics on game word entities.

This exposed a potential bias for us. It appears that in CSE, most of the ideas related to semantics have indeed been largely designed for computers, and not humans as the processing agent.

3.4 RQ4: Precision in Definition

In continuation of RQ1, we wanted to understand the level of definition precision one could expect out of these papers. For instance, if the papers were formal in their content, then we could expect to see formal term definitions for semantics as well.

Our results indicate that while 42 of the 50 were papers had lot of logic and formalisms in them, only 44% (or 22 papers) had definitions for (portions of) semantics. 5 were informal in their definitions. And 3 assumed that semantics were defined elsewhere. So, we could see the pattern that most of the Logic Programming, Formal Methods and Theoretical CSE works perhaps already had a notion of semantics formally defined elsewhere that they could leverage in these documents. And that there very few documents discussing semantics from scratch.

In the case of working with humans and their sense-making of content, no such formal definitions may exist. Therefore, such research would

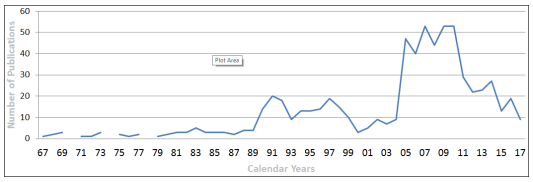


Figure 1: Year-wise histogram of all 653 published papers.

need a more formal definition of semantics – a human oriented semantics – in their publication.

3.5 RQ5: Computer Science Domains

Our goal in RQ5 was to understand which sub areas within Computer Science were actively discussing semantics. Our results indicate that semantics was discussed in multiple sub-areas including: NLP with 3 papers, Programming Languages with 13 papers, Workflows having 2, Databases having 3, Games having 2, Web Services having 5, Logic related papers having 11 and theoretical being 8. Of course, these topics were not mutually exclusive and did overlap. See Figure 2 for a distribution of topics.

The conclusion, therefore, is that notion of semantics is not just restricted to one or two niche areas – like Linguistics, or Programming Languages. There appear to be quite a few emerging areas where semantics – and that too human semantics – can be relevant. For example, mobile web and social web applications has a lot of scope for social and human related content.

3.6 RQ6: Publication History

In RQ6 we wanted to see how hot semantics research has been in the past. We wanted to look at the publication history to draw some context, and from that extrapolate the future outlook for this work.

When we look at the overall corpus of 653 papers, the publications on semantics started in 1967 and continued with just a few publications a year till early 80s. See Figure 1. In the decade of 90s there was a wave of publications for each year contributing to about 10-20 publications each. While 2001-5 was relatively low (with just less than 10 publications a year), the year starting 2005 saw a huge leap in publications: 2006-2011 saw 50-60 publications a year. Starting 2011 to date (2017) we again see a decline in number of papers focused on this topic.

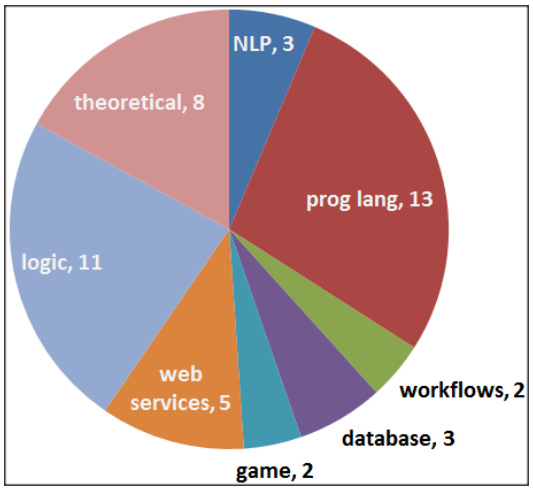


Figure 2: Sub-areas of CSE which published the 50 shortlisted papers.

It appears that semantics had its hayday in the second half of 2005. From 2007 to 20011, that is in the last decade, we could see over 222 publications in this space. However, this number has reduced considerably. In the past five years we could only see 45 publications in this space.

One may interpret this reduction trend to mean that interest in semantics is now waning. However, we take a different stand. We suggest that what is reducing is not interest in semantics, but rather reduction in publications with "definitions of semantics" in them. This could mean two things for us: 1) There is considerable computer oriented, formal definition of semantics already out there that could be leveraged, and 2) there is an excellent opportunity ahead to further define more human oriented semantics for upcoming mobile and web applications.

3.7 Threats to Validity

We recognize that our study sample is (n=)50 and only represents 8% of our excavated corpus of (n=)653. This sample size does indeed effect our results. In addition, we realize that we only focused on papers that used the word "semantic" in their title, or on those that had "definition of semantics" in their body. This also reduces our input corpus.

Broadening our search to also include papers on other related terms could enrich our corpus and through that better inform similar research. But, such resources would come at a cost: They would potentially require more resources in time, effort and reporting. While they may provide more de-

tails, but it may only be marginally different information to the pattern of findings a smaller study could feasibly uncover.

4 Discussion & Insights

From our study we gather that semantics is not merely a study of meaning, but it is study of meaning for humans as well as computers (RQ3), both in natural language as well as in technical languages, both in context (as in usage by a human context) or in context-independent manner (as with lexical analysis in linguistics).

In a computing situation, it appears that there are both 1) theoretical studies that explore the formalism (RQ4), the logic – as in (S18-20,24,32,36 – and 2) application studies, that apply it to web (as in Semantic Web), or to Web Services (S13, 25-26,44), or to Work Flows (S29,49). The theoretical studies tend to be formal and use significant logic (RQ4). Apparently they have contributed to design and development of robust programming languages (S2-3,6,16-17, 21-23,34, 37-38,42) and compilers (S9-11).

In the context of Programming Languages, there is Denotational (S2,12,23,33-35,43), Operational (S23,38,43) and Axiomatic Semantics. Also, the Denotational work was supported with Action Semantics (RQ2).

We also saw that there was application of game theory principles to semantics (S12,46), and, on the other side, application of semantics to graphs (S), images (S30), urban planning (S15), genome studies, databases (S47-48), ontologies (S13), semantic web (S14), web services (S13,25-6,49), Hypertext (S41) etc. Semantics seems to have been used to study similarity (S42-44), distance (S27), tuples, stability. It was applied to systems (S5) as well as for forgetting (S39).

We realize that while the generic term is somewhat ambiguous, in CSE, the term is mostly related to the computer as an processing agent model. Only when it comes to social (as in, biological or urban studies) or web applications level (for example with Ontologies) we found a human interpretation to this term.

From a logic and formalism point of view, semantics has been receiving lot of research attention. However, going forward, there seems to be scope to interpret semantics from the point of view of a human processor. Cognitive Linguistics, Psychology, Information Processing might be able to address the emerging need to make processing as a tool to help the human manage and make sense of the information rendered for her.

5 Conclusions

We undertook the SLR study to systematically explore the notion of semantics, as it is applied in CSE. We presumed that the term Semantic was ambiguously or variedly defined in different subareas of CSE research. What we discovered instead is that the notion of semantics is not ill defined. But, however, it seems to be narrowly defined. Working definitions and application specific definitions seem to exist (S5-6,16,18-19,22-23,27,33-36,38-39,41-44,48-49). Moreover, we found that, in the human computer interaction relationship, most of the focus of the semantics is geared towards the computer being able to process the information(S34,50), to present the information (S41), to access the information (S1,47).

Human semantics (influenced by a computing system) has been, in our opinion, under emphasized (S13,15,46). We see this as an opportunity to develop systems, content and architectures to focus on enhancing *meaning* for the human. No doubt, semantic models and analysis is needed for the back-end computing processor agent. However, such models and analysis should also account for and accommodate a better semantic or easier sense-making ability for the human end user as well. That exploration will be our future work.

References

Serge Abiteboul and Richard Hull. 1987. Ifo: A formal semantic database model. *ACM Trans. Database Syst.*, 12(4):525565.

christiano Braga Alexandre Sztajnberg Alexandre Rademaker. 2005. A rewriting semantics for a software architecture description language.

Sergey S. Gaissaryan Alexey L. Lastovetsky. 2001. An algebraic approach to semantics of programming languages.

Jim Woodcock Andrew Butterfield. 2006. A hardware compiler semantics for handel-c.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Comput. Linguist.*, 36(4):673721.

Johan Benthem. 2005. Guards, bounds, and generalized semantics. *J. of Logic, Lang. and Inf.*, 14(3):263279.

Albert Benveniste, Paul Le Guernic, and Christian Jacquemot. 1991. Synchronous programming with events and relations: the signal language and its semantics. *Science of Computer Programming*, 16(2):103 149.

Ralph Bergmann and Yolanda Gil. 2014. Similarity assessment and efficient retrieval of semantic workflows. *Inf. Syst.*, 40:115127.

Claudio Biancalana, Fabio Gasparetti, Alessandro Micarelli, and Giuseppe Sansonetti. 2013. Social semantic query expansion. *ACM Trans. Intell. Syst. Technol.*, 4(4):60:160:43.

Howard A. Blair. 1982. The recursion-theoretic complexity of the semantics of predicate logic as a programming language. *Information and Control*, 54(12):25 47.

Alexander Bochman. 1998. A logical foundation for logic programming ii" semantics of general logic programs.

Ray Boute. 1988. Systems semantics: Principles, applications, and implementation. *ACM Trans. Program. Lang. Syst.*, 10(1):118155.

Manfred Broy and Christian Lengauer. 1991. On denotational versus predicative semantics. *Journal of Computer and System Sciences*, 42(1):1 29.

Philip T. Cox and Anh Dang, 2010. *Semantic Comparison of Structured Visual Dataflow Programs*, page 11:111:9. VINCI 10. ACM.

Laryssa Machado da Silva, Regina Braga, and Fernanda Campos. 2012. Composer-science: A semantic service based framework for workflow composition in e-science projects. *Inf. Sci.*, 186(1):186208.

Nikos Tzeyelekos Dan R Ghica. 2012. A system-level game semantics.

Stamatia Dasiopoulou, Ioannis Kompatsiaris, and Michael G. Strintzis. 2010. Investigating fuzzy dls-based reasoning in semantic image analysis. *Multimedia Tools Appl.*, 49(1):167194.

Serdar Doğan, Aysu Betin-Can, and Vahid Garousi. 2014. Web application testing: A systematic literature review. *Journal of Systems and Software*, 91:174–201.

Barbara H. Partee Emmon Bach, 2008. *Anaphora and Semantic Structure*, page 122152. Blackwell Publishing Ltd.

Alex Endert, Patrick Fiaux, and Chris North. 2012. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 473–482, New York, NY, USA. ACM.

Sabine Glesner. 2005. A proof calculus for natural semantics based on greatest fixed point semantics. *Electronic Notes in Theoretical Computer Science*, 132(1):73 93. Proceedings of the 3rd International Workshop on Compiler Optimization Meets Compiler Verification (COCV 2004)Compiler Optimization Meets Compiler Verification 2004.

Esfandiar Haghverdi and Philip Scott. 2005. From geometry of interaction to denotational semantics. *Electronic Notes in Theoretical Computer Science*, 122:67 87. Proceedings of the 10th Conference on Category Theory in Computer Science (CTCS 2004)Category Theory in Computer Science 2004.

Bo Stig Hansen Flemming Damm Hans Bruun. 1991. An approach to the static semantics of vdm-sl.

Olaf Owe Jasmin Christian Blanchette. 2008. An open system operational semantics for an object-oriented and component-based language.

Brendan Juba and Madhu Sudan, 2008. *Universal Semantic Communication I*, page 123132. STOC 08. ACM.

Staffs Keele et al. 2007. Guidelines for performing systematic literature reviews in software engineering. In *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*. sn.

Jassin Kessing, Tim Tutenel, and Rafael Bidarra, 2012. *Designing Semantic Game Worlds*, page 2:12:9. PCG12. ACM.

Barbara A Kitchenham, Tore Dyba, and Magne Jorgensen. 2004. Evidence-based software engineering. In *Proceedings of the 26th international conference on software engineering*, pages 273–281. IEEE Computer Society.

Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. 2009. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, 51(1):7–15.

Milos Kravicik and Dragan Gasevic, 2006. *Adaptive Hypermedia for the Semantic Web*, page 310. APS 06. ACM.

Alberto Lavelli, Bernardo Magnini, and Carlo Strapparava, 1992. *An Approach to Multilevel Semantics for Applied Systems*, page 1724. ANLC 92. Association for Computational Linguistics.

Jorge Lobo, Arcot Rajasekar, and Jack Minker. 1991. Semantics of horn and disjunctive logic programs. *Theoretical Computer Science*, 86(1):93 106.

Louis Major, Theocharis Kyriacou, and O Pearl Brereton. 2012. Systematic literature review: teaching novices programming using robots. *IET software*, 6(6):502–513.

Ely Edison Matos, Fernanda Campos, Regina Braga, and Daniele Palazzi. 2010. Celows: An ontology based framework for the provision of semantic web services related to biological models. *J. of Biomedical Informatics*, 43(1):125–136, feb.

Luis Menezes. 2008. Aspect-oriented action semantics descriptions.

David E. Millard, Nicholas M. Gibbins, Danius T. Michaelides, and Mark J. Weal, 2005. *Mind the Semantic Gap*, page 5462. HYPERTEXT 05. ACM.

Chitu Okoli and Kira Schabram. 2010. A guide to conducting a systematic literature review of information systems research.

A. M. Ouksel and A. Sheth. 1999. Semantic interoperability in global information systems. *SIGMOD Rec.*, 28(1):512.

Luca Paolini. 2009. Logical semantics for stability.

Nikolaos S. Papaspyrou. 2001. Denotational semantics of ansi c. *Computer Standards & Interfaces*, 23(3):169 185.

Simon Perdrix. 2008. A hierarchy of quantum semantics.

Fabio Pittarello and Alessandro De Faveri. 2006. Semantic description of 3d environments: A proposal based on web standards. In *Proceedings of the Eleventh International Conference on 3D Web Technology*, Web3D '06, pages 85–95, New York, NY, USA. ACM.

Gollapudi Vrj Sai Prasad, TB Dinesh, and Venkatesh Choppella. 2014. Overcoming the new accessibility challenges using the sweet framework. In *Proceedings of the 11th Web for All Conference*, page 22. ACM.

Gollapudi VRJ Prasad. 2017. Renarrating web content to increase web accessibility. In *Proceedings of the 10th International Conference on Theory and Practice of Electronic Governance*, pages 598–601. ACM.

Franz Puntigam. 1997. Types for active objects based on trace semantics.

Indrakshi Ray, Paul Ammann, and Sushil Jajodia. 1998. A semantic-based transaction processing model for multilevel transactions1. *Journal of Computer Security*, 6(3):181–217.

Daniel M. Russell, Mark J. Stefik, Peter Pirolli, and Stuart K. Card. 1993. The cost structure of sensemaking. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pages 269–276, New York, NY, USA. ACM.

Iván Santiago, Alvaro Jiménez, Juan Manuel Vara, Valeria De Castro, Verónica A Bollati, and Esperanza Marcos. 2012. Model-driven engineering as a new landscape for traceability management: A systematic literature review. *Information and Software Technology*, 54(12):1340–1356.

Robert M. Schwarcz, 1969. *Towards a Computational Formalization of Natural Language Semantics*, page 153. COLING 69. Association for Computational Linguistics.

Kewen Wang Thomas Eiter. 2008. Semantic forgetting in answer set programming.

Ivan Titov and Alexandre Klementiev. 2011. A bayesian model for unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1445–1455, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eran Toch, Avigdor Gal, Iris Reinhartz-Berger, and Dov Dori. 2007. A semantic approach to approximate service retrieval. *ACM Trans. Internet Technol.*, 8(1).

I. V. Velbitskiy. 1977. Metalanguage for formal definition of semantics of programming languages.

V. S. Vykhovanets. 2008. Description of the semantics of context-free languages by the mathematical induction method. 42(4).

Ian Wehrman, David Kitchin, William R. Cook, and Jayadev Misra. 2008. A timed semantics of orc. *Theoretical Computer Science*, 402(23):234 248. Trustworthy Global Computing.

Will Winsborough. 1992. Multiple specialization using minimal-function graph semantics. *The Journal of Logic Programming*, 13(23):259 290.

Jianbo Xu, Zhonglin Xu, and Jiaxun Chen, 2006. *Semantic Retrieval System Based on Ontology*, page 124129. ISP06. World Scientific and Engineering Academy and Society (WSEAS).

Liu Yong, XU Congfu, Pan Zhigeng, and Pan Yunhe. 2004. Semantic modeling project: Building vernacular house of southeast china. In *Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '04, pages 412–418, New York, NY, USA. ACM.

Liangzhao Zeng, Hui Lei, and Badrish Chandramouli, 2005. *Semantic Tuplespace*, page 366381. ICSOC05. Springer-Verlag.

Liangzhao Zeng, Boualem Benatallah, Guo Tong Xie, and Hui Lei, 2006. *Semantic Service Mediation*, page 490495. ICSOC06. Springer-Verlag.

Yi Zhou and Yan Zhang. 2017. A progression semantics for first-order logic programs. *Artificial Intelligence*, 250:58 79.

Semantic Enrichment Across Language: A Case Study of Czech Bibliographic Databases

Pavel Smrz
Brno University of Technology
Faculty of Information Technology
Bozetechova 2, 612 66 Brno, Czechia
smrz@fit.vutbr.cz

Lubomir Otrusina
Brno University of Technology
Faculty of Information Technology
Bozetechova 2, 612 66 Brno, Czechia
iotrusina@fit.vutbr.cz

Abstract

This paper deals with semantic enrichment of textual resources by means of automatically generated named entity recognizers-linkers and advanced indexing and searching mechanisms that can be integrated into various information retrieval and information extraction systems. It introduces a new system transforming Wikipedia and other available sources into task-specific knowledge bases and employs contextual information to build state-of-the-art entity disambiguation components. Although some components are language-dependent (for example, that responsible for the morphology analysis or the semantic role identification), they can be easily replaced by existing tools providing specific functions. As a case study, we demonstrate an instantiation of the system for the task of semantic annotation of Czech bibliographic databases in the context of the CPK project. We particularly stress the role of problem-specific knowledge sources that can be easily integrated into our system and play a key role in the success of the tool in real applications.

1 Introduction

Semantic enrichment of textual resources is a well-studied field with applications in many different domains, such as news, market analysis, environmental studies or cultural heritage. Various general-purpose tools have appeared in recent years (some of them are discussed in the next section). Existing systems can often recognize basic entities and provide a link to a knowledge base (KB), but they usually lack additional information on entities that is critical for specialised applications. Indeed, it is not guaranteed that a referred KB entry contains complete and pertinent information, such as task-relevant entity (sub-)type and attributes.

Moreover, it is almost impossible to re-purpose or re-target existing semantic enrichment systems to a new domain or a new context or to adapt and extend it for another language. Clearly, tools for specific domains call for an integration of specific information sources. For example, given a person, a bibliographical system needs information about publications written by him or her, or a list of publications mentioning the person. Although domain-specific knowledge can substantially improve results of entity disambiguation, it is very difficult to make the existing systems use such information.

The ultimate goal of the work reported in this paper is to build a state-of-the-art semantic enrichment system that will be more flexible than existing tools and will need only a minimal effort to adapt to new environments and tasks. The background KB is derived directly from Wikipedia dumps and domain-specific knowledge sources. Hence, it has always up-to-date information and can benefit from recent updates (as opposed to, e. g., DBpedia-based systems relying on the resource updated twice a year which is usually built from Wikipedia versions several months old).

The rest of this paper is structured as follows. The next section surveys existing systems for semantic enrichment and points out differences in the approach they follow. Section 3 deals with the process of KB creation and compares results of the developed method with alternative solutions. Section 4 describes a case study of the semantic enrichment of Czech texts and bibliographic metadata. Last section concludes our work and discusses future directions of our research.

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 523–532

2 Related Work

The need to bridge the semantic gap between the semi-structured "web of documents" and the structured "web of knowledge" (Buitelaar and Cimiano, 2008) has led to the development of various semantic enrichment systems in recent years. Named entity recognition (NER), linking (NEL) and disambiguation (NED) present a key component of the process of semantic enrichment.

Tools such as DBpedia Spotlight (Daiber et al., 2013), Illinois Wikifier (Ratinov et al., 2011), AIDA (Hoffart et al., 2011), or Babelfy (Moro et al., 2014) enable annotating mentions of named entities in a plain text and "anchoring" the annotations in linked open data resources (most frequently in DBpedia/Wikipedia).

State-of-the-art NED methods have to cope with trade-offs among output accuracy, run-time efficiency and scalability. Fast methods, like Illinois Wikifier or DBpedia Spotlight use relatively simple contextual features. These methods perform well on standard texts that deal with prominent entities, but their accuracy is rather low on more complex inputs with highly ambiguous names. On the other hand, sophisticated systems, such as AIDA or Babelfy, rely on rich contextual features, such as key phrases and joint-inference algorithms. Consequently, they tend to be rather slow.

DBpedia Spotlight is a system for automatically annotating text documents with DBpedia[1] URIs. Its disambiguation algorithm is based on the cosine similaritiy and a modification of TF-IDF weights. The main phrase spotting algorithm relies on exact string matching, which uses LingPipes [2] Aho-Corasick implementation.

Illinois Wikifier combines local clues and global coherence of the joint cross-linking assignment by analysing Wikipedia link structure and estimating pairwise article relatedness. It aims at linking all possible concepts to their corresponding Wikipedia pages.

AIDA employs the YAGO knowledge base[3] as an entity catalogue and a source of entity types and semantic relationships among entities. It uses co-occurrence information obtained from large, syntactically parsed corpora as a similarity measure. The AIDA system provides an integrated NED method using popularity, similarity, and graph-based coherence. AIDA-light (Nguyen et al., 2014) is a lightweight version of AIDA. It is a complete system for NED, which is orders of magnitude faster than AIDA while achieving comparable output quality.

Babelfy provides a unified approach to word sense disambiguation and entity linking. It is knowledge-based and exploits semantic relations between word meanings and named entities from BabelNet[4] (Navigli and Ponzetto, 2012) – a multilingual semantic network consisting of more than 13 million concepts and named entities in 271 languages. It is based on a loose identification of candidate meanings (substring matching instead of exact matching) coupled with a densest subgraph heuristic which selects high-coherence semantic interpretations.

Czech named entity recognition has become a well-established field as well. In (Straková et al., 2016), authors present a completely featureless, language-agnostic named entity recognition system. The system uses only surface forms of words, lemmata and tags as its input. Despite that, it surpasses the precision of current state-of-the-art Czech NER systems, which use manually designed rule-based classification features, such as first character capitalization, existence of special characters in the word, or regular expressions designed to reveal particular named entity types. The system is based on artificial neural networks with parametric rectified linear units, word embeddings and character-level embeddings, which do not need manually designed classification features or gazetteers.

Ni and Florian (2017) proposed a new class of approaches that utilize Wikipedia entity type mappings to improve multilingual NER systems. They apply a maximum entropy classifier on English Wikipedia to construct an entity type mapping. To build multilingual mappings, they use the inter-language links of Wikipedia. The system has a fine-grained entity type set containing 51 types (such as person, organization, location, title work, facility, event, date, time…). They use both – structured information, such as title and infobox, and unstructured information, such as text in a Wikipedia page, as features. The classifier trained with all the features identifies the correct

[1] http://dbpedia.org/

[2] http://alias-i.com/lingpipe/

[3] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

[4] http://babelnet.org/

category for English with overall F1 score of 90.1. Their approach improved the baseline NER systems for English, Portuguese, Japanese, Spanish, Dutch and German.

Various evaluation campaigns have also recently appeared that compare quality of the NE annotation on collected datasets. Initiatives such as NIST TAC KBP[5] – Knowledge Base Population – Entity Discovery and Linking Track (Ji et al., 2015), NEEL[6] – Named Entity rEcognition and Linking Challenge on Microposts (Rizzo et al., 2015), or ERD[7] – Entity Recognition and Disambiguation Challenge (Carmel et al., 2014) rank participating systems based on their overall performance on collections of specific textual fragments (selected sentences, tweets. . .) that had been manually annotated. As the manual dataset preparation is tedious, the provided training and test data is usually limited to few thousands of entity mentions. Developers of NER tools can measure improvements in annotation quality w.r.t. a particular available dataset or they can use specific benchmarking frameworks such as NERD (Rizzo and Troncy, 2011) or Gerbil (Cornolti et al., 2013), embracing several datasets.

3 Knowledge Base Creation

As mentioned above, we aim at creating a domain-specific knowledge-reference store with the highest possible coverage of entities and specific attributes relevant for a task in hand. Due to its limited applicability across languages and across contexts, we cannot simply apply the approach followed by the DBpedia extraction team (Lehmann et al., 2015). Indeed, specific hand-crafted extraction patterns that are often based on Wikipedia features rare in some languages (e.g., Infoboxes in the case of the Czech Wikipedia) are not easily adaptable to our purposes. On the other hand, we do not want to ignore the Wikipedia structure as a key indicator of entity grouping (and types) and rely solely on pure natural language processing (NLP) of entity definitions. The NLP approach is generally difficult to transfer from one language to another and its success hinges on the discipline authors of Wikipedia articles follow when creating the content.

As opposed to manual approaches, our method employs a straightforward learning technique that capitalizes on the most frequent Wikipedia features in each individual language (most frequently, categories and lists a Wikipedia page is assigned to). The learning process involves two steps of the extraction – identification of entities (more precisely, Wikipedia articles / domain-specific source entries referring to a specific entity) and slot filling of attribute values relevant to a particular task. The overall schema of the Knowledge Base Creation component is presented in Figure 1. Let us focus on the initial step first.

The system enables users to specify a basic set of entity types to be recognized (general ones, such as person, location, event, or domain-specific, for example, visual artists related to a particular place). It then expects a set of examples of (prototypical) representatives of each type. It is also possible to extend the input by negative examples, e.g., entities that are known to be often misrecognized as belonging to a given entity type. The positive examples can be either extracted from other sources such as existing lists of entities of interest, or they can be identified by the user. If the entity type exists in a knowledge source in another language in which entities have been already identified, the system can also take advantage of inter-language links and consider all entities of a given type linked to the language as positive examples for the automatic extraction process. Note that inter-language links are treated as features of the automatic technique described and evaluated below so that one can directly see what value they bring as examples of specific entity types.

Even if the initial set of examples is limited, our method can find a significant portion of all entities of the given type. This can be demonstrated by the results in Table 1 showing the numbers of examples necessary to reach the coverage of 90 % or higher for selected types in English Wikipedia. It is clear that less than 20 examples usually suffice to cover almost all entities of particular types so that even fully manual instantiation of the system does not present a tedious job.

If no negative examples are explicitly provided, the system uses entities of all other types as negative examples for a current type, while considering mutual exclusivity constraints that can be specified by the user (all categories are considered mutually exclusive by default). Depending on the complex-

[5] http://nlp.cs.rpi.edu/kbp/

[6] http://www.lancaster.ac.uk/Microposts2015

[7] http://web-ngram.research.microsoft.com/ERD2014/

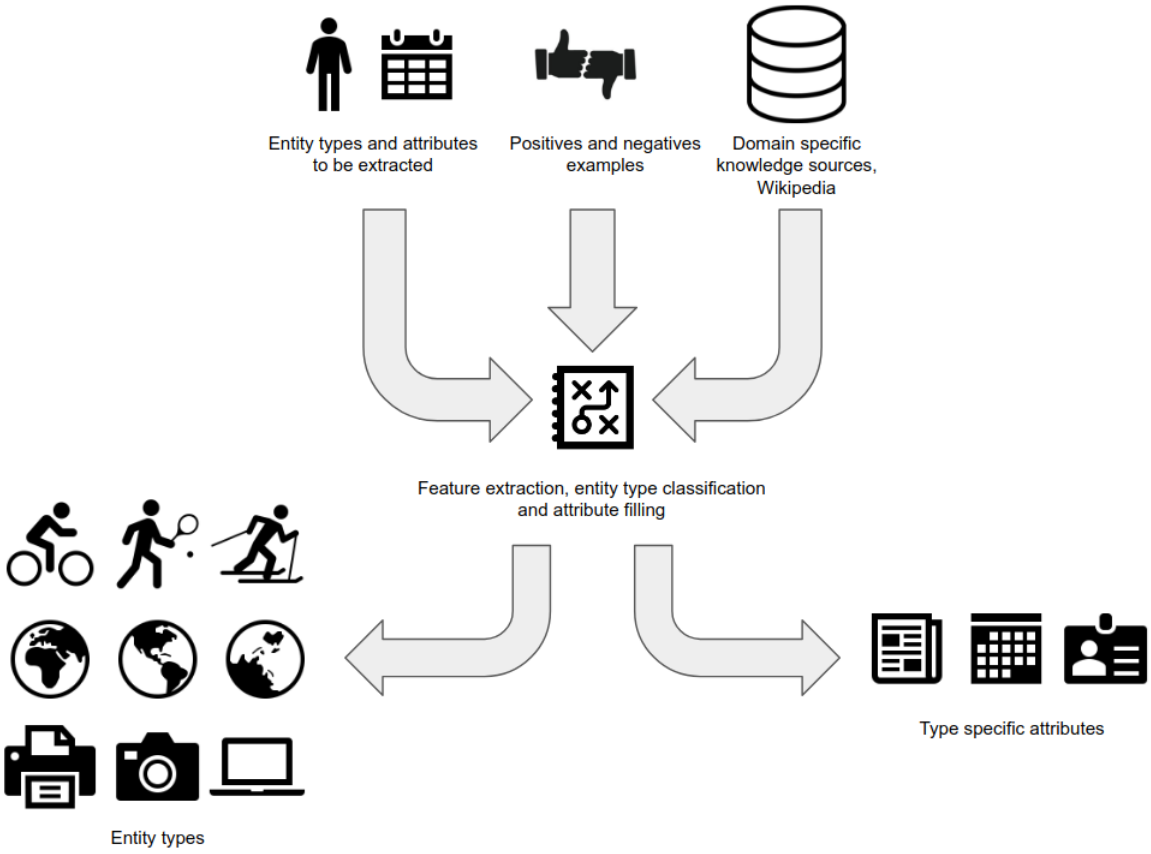


Figure 1: Structure of the KB creation.

Entity type	Min. number of Examples	Coverage
Beverage	10	90.0 %
Musical Work	20	90.3 %
Video Game	15	96.9 %

Table 1: The numbers of examples necessary to reach the coverage of 90 % or higher for selected types in English Wikipedia.

ity of the type, the filtration based on negative examples can bring significant improvements to the automatic extraction.

When dealing with Wikipedia, the learning algorithm explores six types of features:

1. a particular category is assigned to an article;
2. an entity forms a part of a list;
3. a Wikipedia page contains a specific infobox;
4. the key word / phrase of the definition (the first sentence of an article) corresponds to a given list;
5. the first sentence corresponds to a given pattern (for example, an asterisk before a date in parenthesis);
6. the article has an equivalent in a language where the type corresponds to the expected one.

The system generates hypotheses based on provided examples (for example, "all entities in category American Merchants have type Person) and evaluates their relative value. A simple logit model is used to combine the features contributing the most. To find an optimal combination of features, standard measures of the association rule mining are used – support, confidence, lift, and conviction (Bayardo Jr and Agrawal, 1999). Support indicates how frequently an individual feature or a set of features (a feature set) appears in Wikipedia. The confidence measures how often a feature set truly correlates with an entity type, i.e., the proportion of articles corresponding to a feature set F which are known to correspond to type t. The lift is defined as the ratio of the observed support to that expected if F and t were independent. The

conviction can be interpreted as the frequency that the feature set makes an incorrect prediction of the entity type – the ratio of the expected frequency that F occurs with entities of other types than t.

Let symbol $\mapsto$ denote the mapping of a feature set to an entity and E be the set of all entities (or, more precisely, all articles that can deal with an entity). The above-mentioned measures can be formally defined as:

$$\mathrm{supp}(F) = \frac{|\{e \in E; F \mapsto e\}|}{|E|}$$

$$\mathrm{conf}(F \Rightarrow T) = \frac{\mathrm{supp}(F \cup T)}{\mathrm{supp}(F)}$$

$$\mathrm{lift}(F \Rightarrow T) = \frac{\mathrm{supp}(F \cup T)}{\mathrm{supp}(F) \times \mathrm{supp}(T)}$$

$$\mathrm{conv}(F \Rightarrow T) = \frac{1 - \mathrm{supp}(T)}{1 - \mathrm{conf}(F \Rightarrow T)}$$

Table 2 show examples of hypotheses generated by the system for person identification from English Wikipedia along with their support, confidence, lift, and conviction values.

The entity type presents just the initial attribute the automatic system extracts from Wikipedia and/or domain-specific knowledge sources. The user can define a full set of additional attributes corresponding to the entity type that are relevant for a given task. The attributes can reflect the structure of a known template for a given entity type (for example, attributes of a Wikipedia infobox or all potential relations an entity of the type can have in DBpedia). On the other hand, the attribute can be also specific to a given context. For example, our previous work in the cultural heritage domain (Smrz et al., 2013) utilised attributes of art influences and travel experiences of visual artists that could not be directly mapped to a pre-identified relation in DBpedia/Wikipedia infoboxes.

Existing extraction frameworks provide a very limited quality in terms of the completeness of attributes being correctly filled based on information contained in the source. To reach a high coverage and precision, the system applies a learning approach again. Feature detection does not work with the textual content directly. It rather deals with word embeddings (a combination of Word2Vec (Mikolov et al., 2013) and GLOVE (Pennington et al., 2014) vectors) and generalizes the structure of textual fragments corresponding to attribute-filling examples provided by the user.

The resulting knowledge base can be further supplemented by additional information necessary for the actual identification of entities in text, unique identification of entities and their linking to other authoritative knowledge sources, disambiguation information and entity visual representation (if available). As discussed in the following section, entity mentions are matched in text by means of a finite-state technology. The KB should contain all alternative names that can refer to an entity and all forms of names one can expect in the text. For the KB generated from English Wikipedia, we collect and process all alternative names (redirections) and generate known variants of the name forms (for example, shortened or omitted middle names). We also use interlanguage links to record language variants of entity names and consider transcription to characters without diacritics (through Unicode equivalent classes) and transliterations. The generation of name forms in a morphologically rich language (Czech) is discussed in the next section.

We store and later index entity identifiers in their original sources and known interlinks to other LOD (Linked Open Data) resources (for example, Wikipedia URI in both forms with numerical article IDs as well as titles, links to Freebase, DBpedia, etc.). To be able to correctly assign the KB link for an entity with an ambiguous name, we store a vector characterizing words and multiword expressions appearing with an actual meaning of each name. The disambiguation algorithm then reads this data and classifies the entity according to an observed context. If a Wikipedia article referring to an entity contains an image / images, we store this information to be able to represent the entity in a visual way. Figure 2 demonstrates one form of visualization based on such KB.

As already mentioned, the KB resulting from our system covers significantly more entities (correctly assigns the type to more entities) than that of alternative solutions. This is true not only for specific types and less frequent languages such as Czech (see the next section) but also for standard entity types and attributes in English. Table 3 com-

hypotheses	support	confidence	lift	conviction
all entities in category ending with *births* have type Person	23.22	99.51	400.08	15581
all entities whose page contains section *History* don't have type Person	10.17	99.52	132.47	5196
all entities in category ending with *deaths* have type Person	10.79	99.81	401.27	40540
all entities in category starting with *People* have type Person	11.10	99.34	399.40	11536
all entities whose page contains template *coord* don't have type Person	6.48	99.97	133.07	100933

Table 2: Examples of hypotheses for person identification from English Wikipedia.

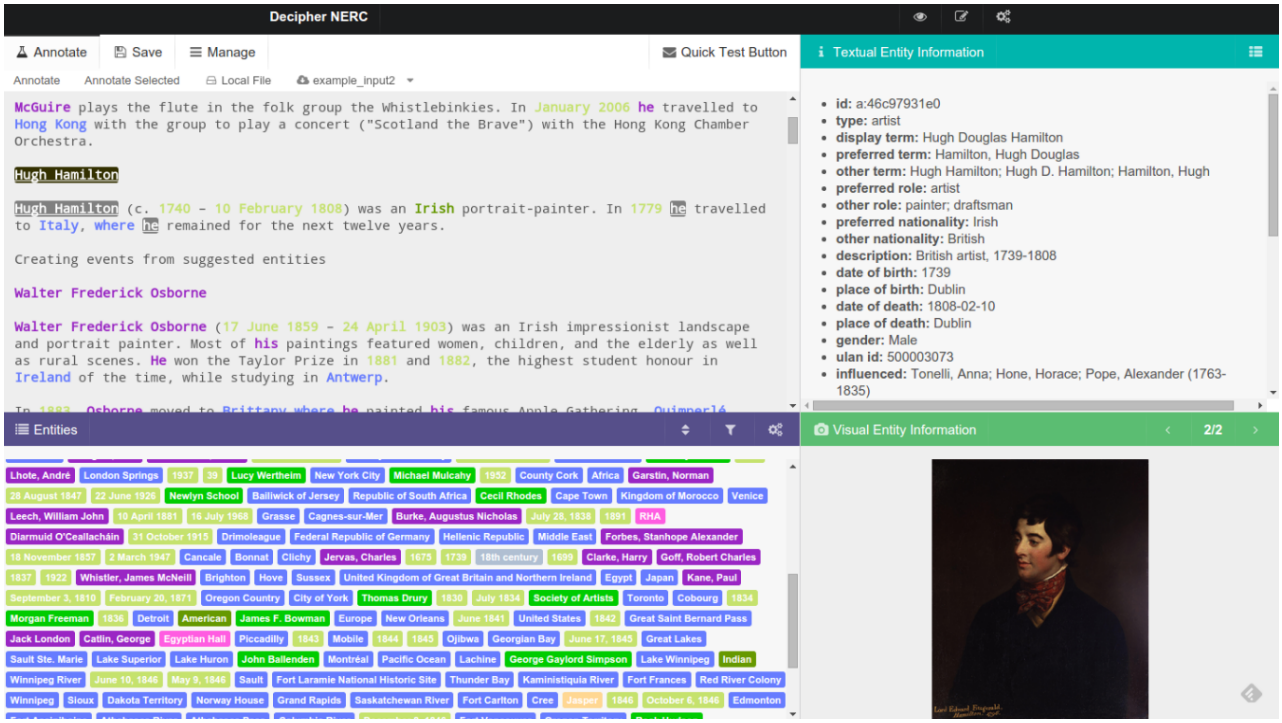


Figure 2: Visualization of an entity.

pares numbers of entities of common general types that can be found using a SPARQL query (often rather a complicated one) in the current DBpedia and the corresponding numbers resulting from our system. It is obvious that if a task requires high precision, it cannot easily rely only on the types identified by DBpedia.

type	number of entities	
	in DBpedia	in our KB
person	1,348,346	1,624,602
place	806,418	738,328
organization	253,215	606,712
video game	20,910	44,862

Table 3: Numbers of entities of common types that can be found using a SPARQL query in the current DBpedia and the corresponding numbers resulting from our system.

4 Semantic Enrichment of Czech Bibliographic Databases

This section presents a case study of the semantic enrichment system applied to Czech bibliographic databases. It forms a part of our work in a large national project – CPK: Using Semantic Technologies to Access Cultural Heritage Through the Central Portal of Czech Libraries. The primary ob-

jective of the project is to research and develop methods leading to a significantly improved access to cultural heritage available in Czech libraries as well as applying results of this research to the creation of the Central portal of Czech libraries (CPK), using advanced semantic technologies. The portal is to be developed in the form of a technological system combining existing and newly developed component applications and a specialized public database (index) of resources integrated into the portal. CPK indexes not only metadata records from contributing institutions, but also full texts of documents digitized by libraries and other information sources.

The fulltexts Czech libraries collect and provide to readers include historical as well as contemporary newspapers and other periodicals, monographs, and various other material of a cultural value. As opposed to bibliographic records associated with monographs that generally contain subject references, including links to authoritative entity knowledge bases (discussed in detail below), the search in the content of newspapers cannot take advantages of the semantically annotated metadata (only a simple fulltext search is supported). That is why our work primary focuses on the available content of periodicals and its automatic semantic enrichment. Note that it is expected that later phases of the project will extend this towards particular categories of monographs (such as historical non-fiction and biographies) and will scale-up the techniques developed for periodicals to a wide range of the cultural heritage artefacts.

Most of the modules integrated into the processing pipeline for the described use case can be easily adapted and used for the same task in any language. We specifically identify the parts that are language-dependent and that need to be replaced by a tool or a resource tailored to the task when the system is to be transferred to another language.

The discussed use case also builds on task-specific knowledge sources. The Czech libraries work with a collection of controlled vocabularies and thesauri, referred to as National Authorities (or simply Authorities), that include named entities (persons, geographical entities, events, etc.). The Authorities are uniquely identified, they provide an official- as well as alternative names, a brief description of the entity, and a link to the Czech Wikipedia (if available). In addition to the information that could be extracted from Wikipedia, the knowledge source can be also used to identify all monographs that deal with particular entities (either in their fulltext form or, at least, as a metadata record listing the title, authors, other subject references, etc.). For the entries corresponding to people, the works authored by a given person can be also identified (currently, only titles and classifications of the work are considered in such cases).

It is critical to recognize that the task-specific knowledge source can bring invaluable quality to the semantic enrichment process and can enable extracting entities and relations not be covered by Wikipedia and other general-purpose resources. This can be demonstrated by the statistics summarized in Table 4. It is obvious that monographs in Czech libraries refer to many entities not addressed by the Czech Wikipedia. To be able to recognize mentions of relevant entities in Czech periodicals and monograph, one cannot simply reckon on Wikipedia only.

A back side of the integration of domain-specific resources lies in a significant increase of ambiguity of names. While there are only 7.1 % of ambiguous person names in the Czech Wikipedia, the knowledge base combining this resource with all Authority files has the ambiguity of 13.3 %. Fortunately, the links to metadata records and fulltexts of monographs dealing with referred Authorities can provide additional learning contexts for the entity disambiguation process. This is also true for entities covered by both Wikipedia and national Authorities so that the additional data actually improves the quality of entity disambiguation models. Indeed, combined Wikipedia pages, referred web links and all available WikiLinks (Otrusina and Smrz, 2016) are not usually a match for an entire book dedicated to a person, location, event, or other type of entities.

Czech is a morphologically-rich (inflectional) language. It is not easy to generate all potential forms in which entity names can appear in text. For example, nouns and adjectives distinguish 7 cases (sometimes with 2 forms per case) in singular and 7 cases in plural and multi-word names of entities come in various forms with complex agreement rules (e.g., genitive phrases with prepositional groups). Moreover, not all parts of multi-word names are capitalized (for example, the Czech Republic would be Česká republika in

type	number of entities in national authorities	covered by Czech Wikipedia
person	629,122	14,758 (2.35 %)
place	29,041	3,386 (11.65 %)

Table 4: Entities from Czech Authorities covered by Czech Wikipedia.

Czech) so that recognition of the boundary of an unknown (new) name can be complicated. The generation of various morphological forms of entity names is clearly language-dependent.

The generation of the name forms was divided into two steps. First, we identified all single-word names and single-string parts of multi-word names that were not covered by an existing morphological database/analyser. We employed a statistical learning method that considers various morphological features (esp. word endings and frequency of similar word paradigms) and estimates the most probable morphological paradigm according to which the word would decline. We then generated all potential wordforms and corresponding morphological tags for single-word names and prepared the same for filtering multi-word combinations by means of group agreement rules.

The second step dealt with multi-word entity names only. Known names served as training examples for statistical name-structure prediction method. It showed to be beneficial to recognize not only the structures necessary for name-form generation, but also to estimate precise meaning of the name components in this phase. The algorithm was trained to distinguish the most probable given names from surnames, titles and name specifiers (such as the younger), numerical components of the names (WWI), etc. This was critical for organizing the names into hierarchies as well as for generating variants with name initials and shortened forms, acronym matching, etc.

The process of semantic enrichment of the Czech text is realized in the form of a pipeline combining various text- and language analysing tools and task-tailored classifiers. The input can be provided as a plain text (for example, the content of historical periodicals obtained by means of OCR) or in the HTML form. We employ CLD3 [8] for language identification (currently filtering out all non-Czech documents) and JustText (Pomikálek, 2013) to identify and eliminate potential boilerplate in HTML Depending on the further planned processing, the text can be tokenized and verticalized in this phase and task-relevant links from the HTML can be stored to be considered later too. To enable linguistically-oriented search in the indexed material, the text can be also lemmatized, PoS tagged and parsed. The relevant tools can be easily plugged-in as well as replaced if necessary. We take advantage of UDpipe (Straka and Straková, 2017) – a linguistic processing pipeline providing results in various languages.

The actual entity recognition and disambiguation is realized by the SEC component developed by our team (Dytrych and Smrz, 2016). The tool is publicly available[9] and can be easily instantiated for other contexts and new languages. It takes the knowledge base (in the form described the previous section) and extracts all potential entity names (a primary reference name, all alternative names, morphological forms, and generated variants). A minimum finite-state automaton is constructed from all the name strings and corresponding indices in the knowledge base by means of an algorithm described in (Daciuk et al., 2000). The SEC recognizer is thus able to associate a textual fragment that corresponds (can correspond) to a name with all potential KB entries it can refer to. Using the information stored in the KB (for example, statistics on Wikipedia article popularity) it can also provide prior probabilities of the potential entity linking. The SEC can also plug-in other entity disambiguation modules that take advantage of additional contextual information stored in the KB. The current version for Czech combines a rule-based disambiguation module implementing strict application-specific restrictions (for example, preferring entities covered by national Authority files) and a learning-based disambiguator trained on all the relevant material available in the CPK project. To enable users to easily extract semantic relations and to search the resource semantically, the SEC engine can also incorporate coreference resolution tools, semantic role labellers and

[8] https://github.com/google/cld3

[9] http://sec.fit.vutbr.cz/sec/

other language processing components.

5 Conclusions and Future Directions

The semantic enrichment system introduced in this paper stresses the easiness of system customization and transfer to other languages. We showed that it is worth to pay attention to preparation of the knowledge base component generated from Wikipedia and domain-specific resources. The presented learning-based extraction system leads to better coverage and generates data directly applicable in the entity recognition and linking task. Moreover, entity types and attributes are not "hardwired", they are fully defined by application needs. Results of the case study on semantic enrichment of Czech texts collected in the CPK project further demonstrate that existing knowledge-reference systems bring a significant value to the semantic enrichment process and that a system based just on general-purpose knowledge sources would not satisfy the need of specific applications.

There are several directions of research we will explore in our future work and apply in the CPK project. We are going to extend interlinks with DBpedia, KBpedia, and other resources and take advantages of established hierarchical structures (ontologies) to initialize the set of prototypical attributes in order to simplify the task of type-specific attribute definition. We will also integrate an advanced classifier of specific entity types and attributes that are not covered by the KB. Less known people, places, events and other entities are often introduced in newspaper texts (usually the first time they appear) so that the system could suggest extensions to the national Authority database adding the most cited new entities. As for the search interface, a lot needs to be done to make the system more intuitive and accessible to laymen.

Acknowledgments

The work reported in this paper has been supported by the Ministry of Culture of the Czech Republic, programme NAKI II, project DG16P02R006 CPK: Using Semantic Technologies to Access Cultural Heritage Through the Central Portal of Czech Libraries.

References

Roberto J Bayardo Jr and Rakesh Agrawal. 1999. Mining the most interesting rules. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 145–154. ACM.

Paul Buitelaar and Philipp Cimiano. 2008. *Ontology learning and population: bridging the gap between text and knowledge*, volume 167. IOS Press.

David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June Paul Hsu, and Kuansan Wang. 2014. ERD'14: Entity recognition and disambiguation challenge. In *ACM SIGIR Forum*, volume 48, pages 63–77. ACM.

Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. 2013. A framework for benchmarking entity-annotation systems. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pages 249–260, New York, NY, USA. ACM.

Jan Daciuk, Stoyan Mihov, Bruce W Watson, and Richard E Watson. 2000. Incremental construction of minimal acyclic finite-state automata. *Computational linguistics*, 26(1):3–16.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*.

Jaroslav Dytrych and Pavel Smrz. 2016. Interaction patterns in computer-assisted semantic annotation of text - an empirical evaluation. In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, pages 74–84. INSTICC, SciTePress.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, Edinburgh, Scotland*, pages 782–792.

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of TAC-KBP2015 tri-lingual entity discovery and linking.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics (TACL)*, 2:231–244.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Dat Ba Nguyen, Johannes Hoffart, Martin Theobald, and Gerhard Weikum. 2014. Aida-light: High-throughput named-entity disambiguation. In *LDOW*.

Jian Ni and Radu Florian. 2017. Improving multilingual named entity recognition with wikipedia entity type mapping. *arXiv preprint arXiv:1707.02459*.

Lubomir Otrusina and Pavel Smrz. 2016. Wtf-lod - a new resource for large-scale ner evaluation. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

J Pomikálek. 2013. justext: Heuristic based boilerplate removal tool. *Available: Google code, online http://code. google. com/p/justext*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics.

Giuseppe Rizzo and Raphaël Troncy. 2011. Nerd: evaluating named entity recognition tools in the web of data.

Giuseppe Rizzo, Amparo E Cano, Bianca Pereira, and Andrea Varga. 2015. #microposts2015 named entity recognition & linking challenge. In *5th International Workshop on Making Sense of Microposts*.

Pavel Smrz, Lubomir Otrusina, Jan Kouril, and Jaroslav Dytrych. 2013. Decipher deliverable D4.3.1: Semantic Annotator. Technical report, Brno University of Technology, Faculty of Information Technology.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada, August. Association for Computational Linguistics.

Jana Straková, Milan Straka, and Jan Hajič. 2016. Neural networks for featureless named entity recognition in czech. In *International Conference on Text, Speech, and Dialogue*, pages 173–181. Springer.

Author Index