ICON 2019

16th International Conference on Natural Language Processing

Proceedings of the Conference

18-21 December 2019
International Institute of Information Technology, Hyderabad, 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 a 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 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 the development of basic tools for multiple languages which are interoperable for machine translation, cross lingual search, handwriting 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 the near future.

This conference proceedings contains papers selected for presentation in technical sessions of ICON-2019 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 88 submissions, 28 papers were selected, 19 for full presentation, 9 for poster presentation, representing a variety of new and interesting developments, covering a wide spectrum of NLP areas and core linguistics. Besides presentations, the conference also hosted 5 tutorials and 1 workshop.

We are deeply grateful to Jan Hajič from Charles University (Czech Republic), C. V. Jawahar from IIIT Hyderabad (India) and Amba Kulkarni from University of 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 cooperation in making ICON 2019 a success.

We thank Anil Kumar Singh, Samar Hussain, Co-Chairs, Workshop/ Tutorial for taking the responsibilities of the events. We are thankful to Radhika Mamidi, Anil Kumar Vuppala and Manish Shrivastava for making the organization of the event at the International Institute of Information Technology (IIIT) a success.

We convey our thanks to Krishna Sireesha S., P V S Ram Babu, G Srinivas Rao, A Lakshmi Narayana, Praveen, Hostel and Administrative staff, 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 heartily express our gratitude to Saumitra Yadav, Pruthwik Mishra, Vandan Mujadia, Nirmal Surange, Prashant Kodali and other team members at IIIT Hyderabad 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-2019, without whose overwhelming response the conference would not have been a success.

December 2019
Hyderabad

Pushpak Bhattacharyya
Dipti Misra Sharma
Rajeev Sangal

Conference General Chair :

Rajeev Sangal, IIIT Hyderabad, India

Program Committee:

Dipti Misra Sharma, IIIT Hyderabad, India (Co-Chair)
Pushpak Bhattacharyya, IIT Bombay and Patna, India (Co-Chair)
Balamurali A R , LIF AIX CNRS
Bharat Ram Ambati , Apple Inc.
Rakesh Balabantaray , IIIT Bhubaneswar
Monojit Choudhury , Microsoft Research
Dipankar Das , Jadavpur University
Amitava Das , Wipro AI Lab
Asif Ekbal , IIT Patna
Sanjukta Ghosh , IIT (BHU)
Vishal Goyal , Punjabi University Patiala
Pawan Goyal , IIT Kharagpur
Harald Hammarström , Uppsala University
Samar Husain , IIT Delhi
C V Jawahar , IIIT Hyderabad
Aditya Joshi , CSIRO
Mitesh M. Khapra , IIT Madras
Parameswari Krishnamurthy , University of Hyderabad
Amba Kulkarni , University of Hyderabad
Malhar Kulkarni , IIT Bombay
Ritesh Kumar , Dr. Bhimrao Ambedkar University, Agra
Niraj Kumar , Conduent Labs India
Anil Kumar Singh , IIT (BHU), Varanasi
Anoop Kunchukuttan , Microsoft AI and Research
Bornini Lahiri , IIT Kharagpur
Sobha Lalitha Devi , Au-KBC Research Centre, Anna University
Gurpreet Lehal , Punjabi University
Radhika Mamidi , IIIT Hyderabad
Yuji Matsumoto , Nara Institute of Science and Technology
Pruthwik Mishra , IIIT, Hyderabad
Vinay Kumar Mittal , IIIT Chittoor, Sri City
Ashutosh Modi , IIT Kanpur
Aditya Mogadala , Saarland University
Vandan Mujadia , IIIT, Hyderabad
Animesh Mukherjee , IIT Kharagpur
Jai Nanavati , Navana Tech
Sudip Kumar Naskar , Jadavpur University
Vasudevan Nedumpozhimana , TU Dublin
Deepak P , Queen's University Belfast

Girish Palshikar , Tata Consultancy Services Limited
Tanvina Patel , Cogknit Semantics
Dripta Piplai , IIT Kharagpur
Bapi Raju , IIIT, Hyderabad
Sai Krishna Rallabandi , Carnegie Mellon University
Paolo Rosso , Universitat Politècnica de València
Atanu Saha , Jadavpur University
Peter Scharf , IIIT, Hyderabad
Raksha Sharma , IIT Roorkee
Ravi Shekhar , Queen Mary University of London
Elizabeth Sherly , IIITM-Kerala
Manish Shrivastava , IIIT Hyderabad
Smriti Singh , Samsung Research UK
Manjira Sinha , IIT Kharagpur
Sunayana Sitaram , Microsoft Research India
Keh-Yih Su , Institute of Information Science, Academia Sinica
Partha Talukdar , Indian Institute of Science
Anil Thakur , IIT (BHU) Varanasi
Ashwini Vaidya , IIT Delhi
Sriram Venkatapathy , Amazon
Samudra Vijaya , IIT Guwahati
Anil Kumar Vuppala , IIIT Hyderabad
Saumitra Yadav , IIIT, Hyderabad

Organizing Committee:

Radhika Mamidi, IIIT Hyderabad
Manish Shrivastava, IIIT Hyderabad
Anil Kumar Vuppala, IIIT Hyderabad

Organized by:

Sponsored by:

Table of Contents

Conference Program

Day 1: Thursday, December 19, 2019

+ 9:00 – 9:30 Inaugural Ceremony

+ 9:30 – 10:30 Keynote I by Prof. Jan Hajič.
Title: *"European Language Technology and Resources Infrastructures"*
Session Chair: Prof. Rajeev Sangal

+ 10:30 – 11:00 Tea break

+ 11:00 – 12:30 **Technical Session I:** Information Retrieval and Extraction
Session Chair: Sobha L Devi

A Deep Ensemble Framework for Fake News Detection and Multi-Class Classification of Short Political Statements
Arjun Roy, Kingshuk Basak, Asif Ekbal and Pushpak Bhattacharyya

Language Modelling with NMT Query Translation for Amharic-Arabic Cross-Language Information Retrieval
Ibrahim Gashaw and H.L Shashirekha

Robust Deep Learning Based Sentiment Classification of Code-Mixed Text
Siddhartha Mukherjee, Vinuthkumar Prasan, Anish Nediyanchath, Manan Shah, Nikhil Kumar and Prajakta Kulkarni

+ 12:30 – 13:30 Lunch

+ 13:30 – 14:30 Keynote II by Prof C. V. Jawahar
Title:*"Data Driven Methods for Multilingual and Multimodal Problems"*
Session Chair: Prof. Pushpak Bhattacharyya

+ 14:30 – 16:00

Technical Session II: Machine Learning in NLP
Session Chair: Asif Ekbal

Robust Text Classification using Sub-Word Information in Input Word Representations.
Bhanu Prakash, Priyank Chhipa, Vivek Sridhar and Vinuthkumar Prasan

Introducing Aspects of Creativity in Automatic Poetry Generation
Brendan Bena and Jugal Kalita

Identification of Synthetic Sentence in Bengali News using Hybrid Approach
Soma Das and Sanjay Chatterji

Technical Session III: Speech Processing
Session Chair: S. R. M. Prasanna

Non-native Accent Partitioning for Speakers of Indian Regional Languages
Guntur Radhakrishna, Krishnan Ramakrishnan and Vinay Kumar Mittal

Autism Speech Analysis using Acoustic Features
Abhijit Mohanta and Vinay Kumar Mittal

+ 16:00 – 17:00 Tea break

+ 16:00 – 17:00 Poster Session
Session Chair: Manish Shrivastava

+ 17:00 – 18:00 NLPAI Meeting

+ 18:30 – 20:00 Cultural Programme

+ 20:30 Onwards Dinner

Day 2: Friday, December 20, 2019

+ 9:00 – 10.30 Keynote-III by Prof. Amba Kulkarni
Title: *"Information Coding in Language:Some insights from Indian Grammatical Tradition"*
Session Chair: Sudeshna Sarkar

+ 10:30 – 11:00 Tea break

+ 11:00 – 13:00

Technical Session IV: Information Retrieval and Extraction
Session Chair: Ashutosh Modi

Multi-linguality helps: Event-Argument Extraction for Disaster Domain in Cross-lingual and Multilingual setting
Zishan Ahmad, Deeksha Varshney, Asif Ekbal and Pushpak Bhattacharyya

A Multi-task Model for Multilingual Trigger Detection and Classification
Sovan Kumar Sahoo, Saumajit Saha, Asif Ekbal and Pushpak Bhattacharyya

Unsung Challenges of Building and Deploying Language Technologies for Low Resource Language Communities
Pratik Joshi, Christain Barnes, Sebastin Santy, Simran Khanuja, Sanket Shah, Anirudh Srinivasan, Satwik Bhattamishra, Sunayana Sitaram, Monojit Choudhury and Kalika Bali

Samajh-Boojh: A Reading Comprehension system in Hindi
Shalaka Vaidya and Hiranmai Sri Adibhatla

Technical Session V: NLP for Low Resource Languages
Session Chair: Bharat Ambati

Incorporating Sub-Word Level Information in Language Invariant Neural Event Detection
Suhan Prabhu, Pranav Goel, Alok Debnath and Manish Shrivastava

A little perturbation makes a difference: Treebank augmentation by perturbation improves transfer parsing
Ayan Das and Sudeshna Sarkar

Sanskrit Segmentation revisited
Sriram Krishnan and Amba Kulkarni

Towards measuring lexical complexity in Malayalam
Richard Shallam and Ashwini Vaidya

+ 13:00 – 14:00 Lunch

+ 14:00 – 15:00 Women in NLP workshop
Session Chair: Manjira Sinha

+ 15:00 –15:30 Industry Interaction Session

+ 15:45 –17:00 **Technical Session VI:** Syntax, Semantics and Discourse
Session Chair: Radhika Mamidi

Building Discourse Parser for Thirukkural
Anita R and Subalalitha C N

DRCoVe: An Augmented Word Representation Approach using Distributional and Relational Context
Md. Aslam Parwez, Muhammad Abulaish and Mohd Fazil

Event Centric Entity Linking for Hindi News Articles: A Knowledge Graph Based Approach
Pranav Goel, Suhan Prabhu, Alok Debnath and Manish Shrivastava

+ 17:00 – 17:15 Tea break

+ 17:15 – 17:45 Valedictory Function

Robust Text Classification using Sub-Word Information in Input Word Representations

Mahanti Bhanu Prakash, Priyank Chhipa, Vivek Sridhar, Vinuthkumar Prasan
Samsung R&D Institute India, Bangalore
{me.prakash, p.chhipa, v.sridhar, vinuth}@samsung.com

Abstract

Word based deep learning approaches have been used with increasing success recently to solve Natural Language Processing problems like Machine Translation, Language Modelling and Text Classification. However, performance of these word based models is limited by the vocabulary of the training corpus. Alternate approaches using character based models have been proposed to overcome the unseen word problems arising for a variety of reasons. However, character based models fail to capture the sequential relationship of words inherently present in texts. Hence, there is scope for improvement by addressing the unseen word problem while also maintaining the sequential context through word based models.

In this work, we propose a method where the input embedding vector incorporates sub-word information but is also suitable for use with models which successfully capture the sequential nature of text. We further attempt to establish that using such a word representation as input makes the model robust to unseen words, particularly arising due to tokenization and spelling errors, which is a common problem in systems where a typing interface is one of the input modalities.

1 Introduction

Recent research has demonstrated the success of word based models for NLP problems like Machine Translation (Sutskever, 2014) and Text Classification (Mikolov, 2010). It is well established in literature that the dictionary of words contained in the training corpus have significant bearing on the performance of these models. For example, in the case of language modelling, an unseen word can never be predicted and models also tend to have lower accuracies when predicting words in the vicinity of an unseen word. Models for text classification also suffer from a similar problem wherein one or more unseen words in the input may significantly increase classification error. Alternate approaches using character based models (Zhang, 2015; Kim, 2016) have been proposed to overcome the unseen word problem which ails word-based deep learning networks. However, character based models fail to capture the sequential relationship of words inherently present in texts.

The main contribution of this paper is to establish the suitability and robustness of an input embedded vector which incorporates sub-word information (Bojanowski, 2016) with a recurrent neural network model for sentence classification and also establish the capability of such a configuration to deal effectively with the unseen word problem, especially arising due to word segmentations and spelling errors.

2 Related Work

The input to text based deep learning models is usually a numeric vector representation of text, commonly called embedded vectors. The embedded vector of each word is designed to be indicative of its semantic relationship with other words or characters as available in the corpus in embedded space. This usually constitutes the very first layer of the network. This layer may be initialized randomly or with pre-trained vectors The pre-trained vectors may be static or may also be learned with the network. These pre-trained vectors are typically generated from a large training corpus which is usually not directly related to the problem at hand, but is representative of language as a whole. One of the most commonly

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 1–8
Hyderabad, India, December 2019.

used pre-trained embedding is proposed by Mikolov (2010) where a neural network approach is used to generate word vectors based on a 1.6 billion words data set. An alternate approach discussed in Pennington (2014) focusses on whether distributional word representations are best learned from count-based methods rather than prediction-based methods. Since these vectors are pre-trained using large corpora, they contain meaningful semantic representations of even words not seen in the training corpus for the specific problem, which helps deal with the unseen word problem to a certain extent. In addition, models tend to converge faster when pre-trained vectors are used. However, pre-trained word embedding approaches continue to have difficulty with words not in the dictionary of the input embedding. Also, rare words are often not represented as well as more frequently occurring words. Words not seen in the training corpus are usually either marked as unknown (UNK) or excluded altogether from the input. To address these drawbacks, several alternate approaches have been proposed. Zhang (2015) proposed a 9-layer character based CNN model which addresses the unseen word problem in Word based models. However, this CNN based approach fails to capture sequence context features in the text. Kim (2016) proposed an architecture in which the character embedding is input to a CNN, the output of which acts as input to an RNN. In such models, the CNN component captures the n-gram features of text and RNN takes care of sequence context of such features in the text. For character based CNN models, the context or relationship between multiple characters and words are captured by convolution filters or kernels. A set of fixed filter sizes (n-grams) may not completely capture word-level information. Also, capturing longer context is difficult in CNN models. Another alternate approach is proposed by Bojanowski (2016) where each word is represented as a bag of character n-grams and a vector representation is associated to each character n-gram. Words are represented as a sum of these representations. This is found to be especially effective when dealing with morphologically rich languages. This has been used with shallow models for sentiment analysis and tag identification problem in Joulin (2016). However, for more complex problems over a larger number of classes, the higher representational power of deep networks such as RNNs and CNNs may be desired.

In this work, we apply the method for generating vector representations proposed in Bojanowski (2016) to deep learning networks such as the architecture proposed in Sutskever (2014) and explore the extent to which unseen word problem, especially arising due to misspellings and tokenization errors, is addressed.

3 Proposed Approach

Unseen words are a common occurrence in NLP problems and arise from a variety of situations. The most common reasons for unseen words is simply a lack of exhaustive training data for a specific problem. This problem is largely dealt with by using pre-trained distributions trained on a large corpus. Another common source of unseen words is morphological variance. This is a scenario where the unseen word is close to a seen word both superficially and semantically. Research described in Bojanowski (2016) and Joulin (2016) show that input vectors incorporating sub-word information have proved effective in tackling this problem.

Another source of unseen words are misspellings or incorrect word segmentation. This is a common problem faced in multi-modal applications such as voice assistants wherein one of the input modalities is a typing interface. These types of errors seem to be similar to the UNKs arising from morphological variance wherein the unseen word shares a close superficial as well as semantic similarity with a seen word.

We propose to use pre-trained embedded vectors to deal with the unseen word problem, especially due to misspellings, using word vectors which incorporate sub-word information. An RNN based sentence classifier with an architecture similar to the one proposed by Sutskever (2014) is used and compared with the performance of the distributions described by Mikolov (2013) and Pennington (2014) on standard data sets for text classification. In addition, we intend to simulate the UNK problem due to misspellings and incorrect tokenization by applying rules to the standard data sets. These rules consist of common misspellings such as “ei” instead of “ie”, incorrect double consonants (‘aggressive’ vs ‘aggresive’) and so on.

3.1 Word Representations with Sub-Words

Word vectors are traditionally used to cluster words with a high degree of semantic similarity. These representations help to deal with rare word problem wherein words which are less frequently present in training data are not learnt as well as words which are semantically similar but present more frequently. Word distributions learnt on larger corpora alleviate this problem. Another source of rare or unseen words is morphological variance. This problem is particularly important for languages which have a high degree of inflection. The work by Bojanowski (2016) uses a word representation which incorporates sub-word information to construct word vectors to deal with this problem.

Misspelled words also share several sub-words with the actual word and we therefore propose that a word representation which incorporates sub-word information should be good at dealing with UNKs due to spelling or tokenization errors. This intuition is validated by the clusters shown in the word representations of the misspelling words projected in two dimensions.

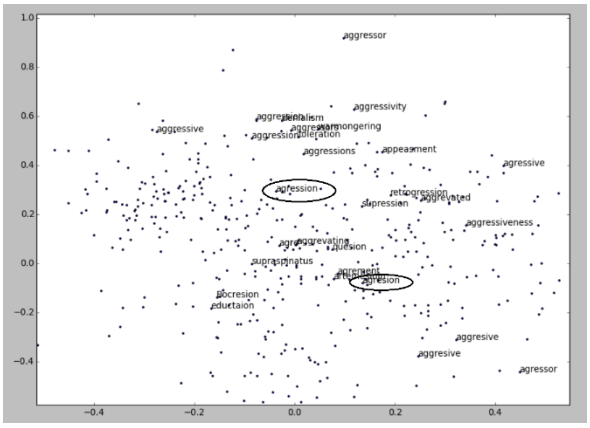


Figure 1: Word clusters for ‘aggression’

As seen in Figure 1, various common misspellings of the word ‘aggression’ are clustered close to the actual word and related words like ‘aggresion’, ‘aggressive’ and so on.

Figure 2 shows that another common misspelling, ‘ie’ instead of ‘ei’, is dealt with well by the word vectors constructed using sub-words.

Figure 3 shows the incorrect tokenization of the phrase ‘remind me’ as ‘remindme’ which is placed in the relevant cluster containing ‘reminds’, ’remind’, ’reminded’ and other similar words.

The above plots indicate the robustness of a word vector built using sub-words to deal with spelling and word segmentation errors. We attempt

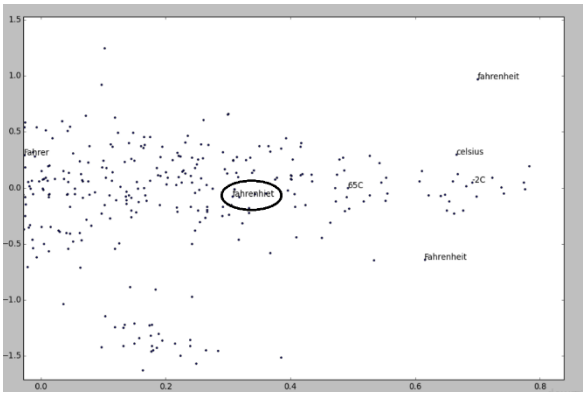


Figure 2: Cluster for 'Fahrenheit'

to establish this conclusively by applying such a word representation to a sentence classification problem using a standard word-based RNN architecture described in the following sections.

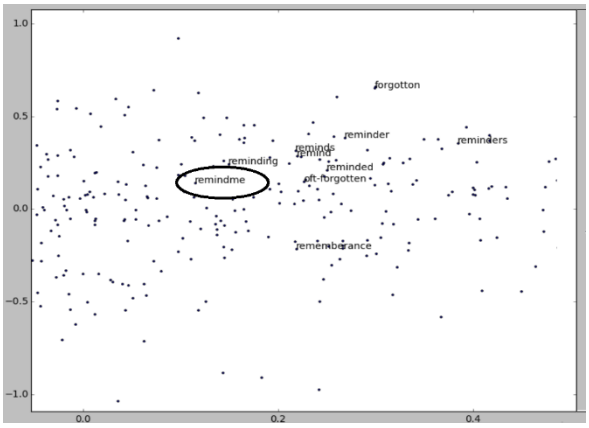


Figure 3: Cluster for tokenization (remind me)

3.2 Sentence Classification RNN Model

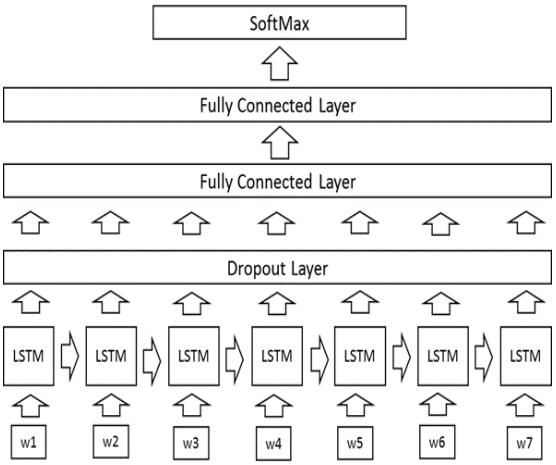


Figure 4: RNN based Sentence Classifier

Sentence classifier architecture used is depicted in Figure 4 and is based on the sequence to

sequence learning architecture for machine translation described in Sutskever (2014).

The input 'wi' is the embedded vector used to represent words in the input text. Experiments were carried out using the word distributions described in Mikolov (2013) and Pennington (2014) and used as reference for comparison.

$$Loss = -\sum_{c=1}^{M} y_{o,c} \log(P_o, c) \quad (1)$$

The model is trained with the 20% dropouts and categorical cross entropy loss function represented by the Equation 1. The rmsprop optimizer is used which is a popular choice for Recurrent Neural networks. A 300-dimension word vector is used as the input.

3.3 Sentence Classification RNN Model with Pre-Trained Embedding containing Sub-Word Information

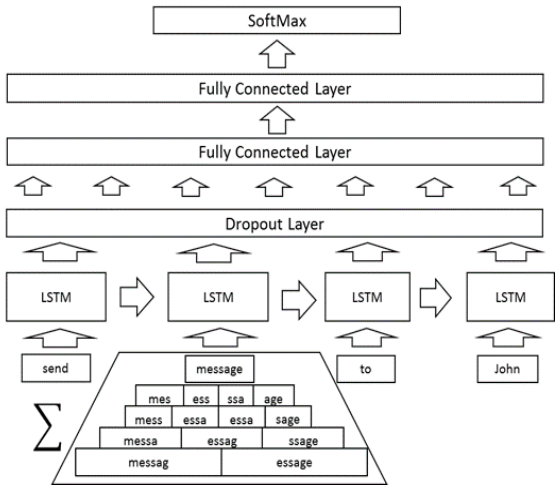


Figure 5: RNN based sentence classifier with sub-word information based embedding

Figure 5 shows the architecture used with the input vector built using a combination of sub-words. One of the configurable parameters while generating the pre-trained embedded vectors is the sub-word length to be incorporated. Sub-words consisting of lengths from 3 to 6 along with the whole word are used to generate the input embedding.

The skip-gram model described by Bojanowski (2016) is used to generate the pre-trained input vector representation using One-billion-word benchmark (Chelba, 2013). This is used for comparison with the reference word distributions described in Mikolov (2013) and Pennington (2014) which are commonly used with deep neural network based architectures.

4 Datasets and Experimental Setup

The sentence classification task is chosen for the work described in this paper.

In our first set of experiments, we apply the reference word-based sequence learning architecture of Sutskever (2014) to the sentence classification problem on three standard datasets. The pre-trained embedding learnt through the method proposed in Bojanowski (2016) is used as a static input embedding and is not updated as part of the training for the specific problem.

We compare the performance of these sub-word based input embedded vectors with the popular GloVe and Word2vec pre-trained word representations using this architecture. This is used to establish the suitability of the word representations of Bojanowski (2016) to deep learning architectures.

In our second set of experiments, we apply the reference word-based sequence learning architecture of Sutskever (2014) to the sentence classification problem on three standard datasets which are modified to incorporate misspellings. Two standard misspelling dictionaries are used to generate the misspelled versions of the standard datasets. This is done in order to simulate real-world situations, such as multi-modal smart assistants, where the input to a sentence classification system may be via a textual input interface, and therefore prone to misspellings. The performance of sub-word based word vectors Bojanowski (2016) is compared with reference distributions. This is used to establish that word representations which are constructed using sub-word information are robust and more suitable for use in multi-modal commercial applications than the more popular GloVe and Word2vec word representations. The datasets used for these experiments are described in detail in the following sections.

4.1 Sentence Classification Datasets

The following three datasets are used for benchmarking on the sentence classification problem.

The SUBJ Subjectivity dataset is a two-class dataset where the task is to classify a sentence as subjective or objective. There are 10000 sentences,

we used 5-fold cross validation with 80% of the data for train and the remaining 20% are used as a test set.

The MPQA Opinion polarity dataset has 10606 sentences with two classes. We use 80% of the data for model training and remaining 20% for testing. A 5-fold cross validation result is presented since a pre-defined test and train split is not available.

AG News dataset is a four-class dataset where the task is to classify the sentence into 'world', 'sports', 'business', 'science and tech'. This dataset has two parts: title and description, but we only considered the title for text classification. The dataset contains 120000 training and 7600 test samples.

4.2 Spelling Error Dictionaries

There are two broad sources of misspellings, namely phonetic and typographic. We use the following two reference dictionaries which focus on these two kinds of misspellings.

The Wikipedia[1] misspelling dictionary contains 2,455 misspellings of 1,922 words. This is a list of common misspellings made by Wikipedia editors. This dictionary focuses mainly on the typographic misspellings, but also includes several common phonetic based spelling errors.

The Aspell[2] dictionary contains 531 misspellings of 450 words. This dataset focusses on phonetic misspellings. Aspell begins by converting the misspelt word to its sounds-like equivalent using Metaphone and moves on to find all words that have a sounds-like within one or two edit distances from the original word's sounds-like. These sounds-like words are the basis for the suggestions of Aspell. This is derived by Atkinson[2] for testing the GNU Aspell spellchecker.

5 Results

The reference model of Sutskever (2014) described in the sections above with GloVe and Word2vec embedding vectors used as the input word vectors has been compared with the proposed word embedding on the three standard sentence classification datasets as described in the Table 1.

The main purpose of this experiment is to prove the performance of the proposed word embedding using sub-words with a word-based RNN sequence learning architecture.

[1] http://www.dcs.bbk.ac.uk/~ROGER/wikipedia.dat

[2] http://aspell.sourceforge.net/

The performance of the proposed input word embedding applied to the reference architecture is

Dataset	Test Set Size	Standard Data Set		
		GloVe	Word2vec	Sub-word embedding
subj	1000	85.68	86	84.58
MPQA	1000	89.8	89.8	89.8
AGNews	7600	87.2	87.5	87.5

Table 1: Comparison of reference distributions with proposed approach on standard datasets

comparable to the performance with the Pennington (2014) and Mikolov (2010) input embedded vectors. This illustrates the suitability of the sub-word embedding for use with deep neural networks.

The accuracies shown in Table 1 above are used as benchmarks for our further investigation into the capacity of the various types of input embedded vectors to deal with misspellings and tokenization errors.

Dataset	Misspelling Dictionary : Wikipedia			
	Changed Data	GloVe	Word2vec	Sub-word embedding
subj	964/1000	79.72	78	81.3
MPQA	401/1000	66.62	66.66	87.3
AGNews	2201/7600	83.3	82.7	86.3

Table 2: Comparison of reference distributions with proposed approach on standard datasets with misspellings from Wikipedia

The Wikipedia misspelling dictionary mainly focusses on typographic misspellings. The different datasets are also differently prone to spelling errors. In the case of Subj dataset, 964 out of 1000 test sentences are modified, but a majority of these misspellings are words like 'the' and 'and', which are typically less likely to affect the classification result. However, 40% of the MPQA test set is modified and about 29% of the AG News data set is modified by the Wikipedia misspelling dictionary. This serves to illustrate the need to deal with misspellings as part of any commercial application.

The results in Table 2 show that the proposed approach is always better than the reference embedded vectors at dealing with the UNK problem arising due to spelling errors. In certain cases, the improvement is marginal (Subj: ~1%)

whereas in the best case (MPQA), a huge improvement of over 20% is observed. These results indicate that the proposed approach is significantly better at dealing with UNKs arising due to typographic misspellings.

The Aspell misspelling dictionary mainly focusses on phonetic misspellings. Similar to the discussion above, the different datasets show varying susceptibility to spelling errors. Applying the misspellings from the Aspell dictionary results in 12% and 14% of the MPQA and AG News test sets being modified respectively. In the case of the Subj dataset, a much larger 86% of the 1000 test sentences are modified.

Dataset	Misspelling Dictionary : Aspell			
	Changed Data	GloVe	Word2vec	Sub-word embedding
subj	859/1000	81.75	80.56	83.8
MPQA	145/1000	70.69	70.6	89
AG News	970/7600	84.5	83.3	86.79

Table 3: Comparison of reference distributions with proposed approach on standard datasets vs misspellings from Aspell

The results in Table 3 show that GloVe and Word2vec word representations show a drop in performance due to misspellings for all 3 datasets ranging from 3% in the case of AG News to 19% in the case of MPQA dataset and that the proposed approach is better than the reference embedded vectors at dealing with the UNK problem arising due to phonetic spelling errors in all cases.

Datasets	Best Accuracy (without misspelling)	Proposed Approach (Wikipedia misspelling)	Proposed Approach (Aspell misspelling)
SUBJ	86	81.3	84.58
MPQA	89.8	87.3	89
AG news	87.5	86.3	86.79

Table 4: Comparison of proposed approach on misspelled data with best accuracy on original datasets

Table 4 compares the performance of the proposed approach on the datasets modified with misspellings with the best accuracy out of any of the three word representations on the original dataset without misspellings. The purpose of this comparison is to measure the extent to which the proposed approach addresses the problem of spelling errors.

In most of the cases, only a minor drop in accuracy is observed ranging from 0.7% to 2.4%. The only outlier is the Wikipedia dictionary modified SUBJ dataset where a significant drop of over 4.7% is seen. Detailed analysis shows that the most common spelling modifications in this dataset are the words 'the' and 'and' which the sub-word based representation doesn't deal with well as the number of sub-words for very short words are too less to have a significant impact in generating the word vectors. Some more specific situations which are not handled well by the proposed approach are discussed in the following section along with the direction our future work will take to address these problems.

Overall, the performance of the proposed approach is close enough to the performance on the original datasets without misspellings to indicate that the proposed approach is not only comparable to the state-of-the-art when applied to deep learning architectures but also solves the UNK problems arising due to typographic and phonetic misspellings to a significant extent.

6 Discussion

It is seen that 7% of the attendees of the TOEFL[3] examination, a test of English, tend to make spelling errors, even in an environment where the sole focus is correctness of grammar and language. Our study of internal data from a Voice Assistant applications indicates that in excess of 25% of all data input using a typing interface contains errors in spelling and word breaks. This illustrates the need for a method to handle spelling errors gracefully and reliably, especially for more natural AI applications.

The major motivation to conduct the investigations presented in this work was to come up with a technique to deal with the misspelling problem which is inherently present in multi-modal voice assistants where the primary input paradigm is speech, which is not prone to misspellings at all, and the secondary modality is a typing interface which is quite prone to spelling and word segmentation errors. The goal was to use a technique wherein the models trained on well-formed data are robust to errors in spelling rather than to implement a relatively clumsy rule-based preprocessing module which would attempt to

[3] https://www.researchgate.net/figure/Average-percent-of-misspelled-words-per-essay-by-NS-NNS-and-score-panel-A-GRE-data_fig3_277584335

correct misspellings but would tend to be unreliable by nature.

A study of the misspelling dictionaries and the substitutions made to the standard datasets using these dictionaries gives further clarity on the various types of spelling errors commonly seen. The first level classification of the types of spelling errors is typographical and phonetic. Typographical errors consist mainly of omission, addition or swapping of characters. All these three cases seem to be handled reasonably well using the sub-words approach to construct word vectors. The second major category of phonetic misspellings consists mainly of replacement of characters by other similar characters such as 'destruction' vs 'distruction' and so on. Other common errors of this kind are incorrect usage of double consonants, 'ei' instead of 'ie' and so on. The majority of these cases are also handled well by the sub-word based word representations. One of the observations while analyzing the drop in accuracy of the models on the misspelled datasets is that misspellings towards the middle of the word are not dealt with as well as misspellings near either end of the words since more number of sub-words are affected in this case. We are currently working on some improvements to the word representations to overcome this problem.

7 Conclusion

The work in this paper demonstrates that input embedded vectors which incorporate sub-word information and are learnt through a shallow network are well-suited for use with sequence aware deep learning networks. It also showcases the effectiveness of such a configuration in dealing with various common types of spelling errors arising due to both typographic as well as due to phonetic reasons. The results showing that the accuracy of the proposed configuration on the standard datasets with misspellings is comparable to the best performance on the misspelling free datasets indicate that the proposed configuration almost entirely solves the problem of spelling errors.

The proposed work is especially suited for use is multi-modal applications as it not only seamlessly handles spelling errors but performs as well as state-of-the-art systems on correctly spelled inputs. One example of such a real-world application is a multi-modal voice assistant which allows textual input in addition to speech input. Modern multi-modal voice assistants attempt to support a very wide range of complex functionality for which deep learning networks are a natural choice and will greatly benefit from an input embedding which seamlessly handles misspellings. Moreover, the approach used to construct these input embedded vectors also handles morphological variance and is applicable across languages.

This work also establishes the similarity in nature between morphological variance and spelling or tokenization errors wherein the unseen word is both semantically and superficially similar to an actual seen word, and therefore improvements made in dealing with one are likely to be beneficial in dealing with the other. This opens up the possibility of a wide area of research as this work proves a significant overlap between two problem statements which were hitherto perceived to be different.

8 Future Work

Our future work will focus on proving the applicability of the proposed approach across languages by extending the experiments conducted here to more languages.

Another line of research we are pursuing focusses on improving the method of selecting sub-words in order to deal better with certain kinds of morphological variance and spelling errors, such as omission of a character in the middle of a long word, which this proposed approach doesn't deal with well in some cases.

We also intend to improve this approach to be robust to other variance arising from other forms of textual input such as text messages, tweets and so on.

References

Bojanowski, P. et al., 2016. Enriching word vectors with subword information. *Computing Research Repository*, arXiv:1607.04606. Version 2.

Chelba, C. et al., 2013. One billion word benchmark for measuring progress in statistical language modeling. *Computing Research Repository*, arXiv:1312.3005. Version 3.

Kim, Y. et al., 2016. Character-Aware Neural Language Models. *Association for the Advancement of Artificial Intelligence*, pages 2741-2749

Kim, Y. 2014. Convolutional neural networks for sentence classification. *Computing Research Repository*, arXiv:1408.5882. Version 2.

Joulin, A. et al., 2016. Bag of tricks for efficient text classification. *Computing Research Repository*, arXiv:1607.01759. Version 3.

Mikolov, T. et al., 2010. Recurrent neural network based language model. *Eleventh Annual Conference of the International Speech Communication Association*.

Mikolov, T. et al., 2013. Efficient estimation of word representations in vector space. *Computing Research Repository*, arXiv:1301.3781. Version 3.

Pennington, J. et al., 2014. Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532-1543.

Sutskever, I. et al., 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, pages 3104-3112.

Zhang, X. et al., 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, pages 649-657.

A Deep Ensemble Framework for Multi-Class Classification of Fake News from Short Political Statements

Arjun Roy, Kingshuk Basak, Asif Ekbal, Pushpak Bhattacharyya
Department of Computer Science and Engineering,
Indian Institute of Technology Patna
{arjun.mtmc17, kinghshuk.mtcs16, asif, pb} @iitp.ac.in

Abstract

Fake news, rumor, incorrect information, and misinformation detection are nowadays crucial issues as these might have serious consequences for our social fabrics. Such information is increasing rapidly due to the availability of enormous web information sources including social media feeds, news blogs, online newspapers etc. In this paper, we develop various deep learning models for detecting fake news and classifying them into the pre-defined fine-grained categories. At first, we develop individual models based on Convolutional Neural Network (CNN), and Bi-directional Long Short Term Memory (Bi-LSTM) networks. The representations obtained from these two models are fed into a Multi-layer Perceptron Model (MLP) for the final classification. Our experiments on a benchmark dataset show promising results with an overall accuracy of 44.87%, which outperforms the current state of the arts.

1 Introduction

> "We live in a time of fake news-things that are made up and manufactured." Neil Portnow.

Fake news, rumors, incorrect information, misinformation have grown tremendously due to the phenomenal growth in web information. During the last few years, there has been a year-on-year growth in information emerging from various social media networks, blogs, twitter, facebook etc. Detecting fake news, rumor in proper time is very important as otherwise, it might cause damage to social fabrics. This has gained a lot of interest worldwide due to its impact on recent politics and its negative effects. In fact, Fake News has been named as 2017's word of the year by Collins dictionary[1].

Many recent studies have claimed that US election 2016 was heavily impacted by the spread of Fake News. False news stories have become a part of everyday life, exacerbating weather crises, political violence, intolerance between people of different ethnics and culture, and even affecting matters of public health. All the governments around the world are trying to track and address these problems. On 1^{st} Jan, 2018, bbc.com published that "Germany is set to start enforcing a law that demands social media sites move quickly to remove hate speech, fake news, and illegal material." Thus it is very evident that the development of automated techniques for detection of Fake News is very important and urgent.

1.1 Problem Definition and Motivation

Fake News can be defined as completely misleading or made up information that is being intentionally circulated claiming as true information. In this paper, we develop a deep learning based system for detecting fake news.

Deception detection is a well-studied problem in Natural Language Processing (NLP) and researchers have addressed this problem quite extensively. The problem of detecting fake news in our everyday life, although very much related to deception detection, but in practice is much more challenging and hard, as the news body often contains a very few and short statements. Even for a human reader, it is difficult to accurately distinguish true from false information by just looking at these short pieces of information. Developing suitable hand engineered features (for a classical supervised machine learning model) to identify fakeness of such statements is also a technically challenging task. In contrast to classical feature-based model, deep learning has the advantage in

[1] http://www.thehindu.com/books/fake-news-named-word-of-the-year-2017/article19969519.ece

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 9–17
Hyderabad, India, December 2019.

the sense that it does not require any handcrafting of rules and/or features, rather it identifies the best feature set on its own for a specific problem. For a given news statement, our proposed technique classifies the short statement into the following fine-grained classes: *true*, *mostly-true*, *half-true*, *barely-true*, *false* and *pants-fire*. Example of such statements belonging to each class is given in Table 1 and the meta-data related to each of the statements is given in Table 2.

1.2 Contributions

Most of the existing studies on fake news detection are based on classical supervised model. In recent times there has been an interest towards developing deep learning based fake news detection system, but these are mostly concerned with binary classification. In this paper, we attempt to develop an ensemble based architecture for fake news detection. The individual models are based on Convolutional Neural Network (CNN) and Bi-directional Long Short Term Memory (LSTM). The representations obtained from these two models are fed into a Multi-layer Perceptron (MLP) for multi-class classification.

1.3 Related Work

Fake new detection is an emerging topic in Natural Language Processing (NLP). The concept of detecting fake news is often linked with a variety of labels, such as misinformation (Fernandez and Alani, 2018), rumor (Chen et al., 2017), deception (Rubin et al., 2015), hoax (Tacchini et al., 2017), spam (Eshraqi et al., 2015), unreliable news (Duppada, 2018), etc. In literature, it is also observed that social media (Shu et al., 2017) plays an essential role in the rapid spread of fake contents. This rapid spread is often greatly influenced by social bots (Bessi and Ferrara, 2016). It has been some time now since AI, ML, and NLP researchers have been trying to develop a robust automated system to detect Fake/ Deceptive/ Misleading/ Rumour news articles on various online daily access media platforms. There have been efforts to built automated machine learning algorithm based on the linguistic properties of the articles to categorize Fake News. Castillo et al. (2011) in their work on social media (twitter) data showed that information from user profiles can be useful feature in determining veracity of news. These features were later also used by Gupta et al. (2014) to build a real-time system to access credibility of tweets using SVM-rank. Researchers have also attempted to use Rule-Based and knowledge driven techniques to track the problem. Zhou et al. (2003) in their work showed that deceptive senders have certain linguistic cues in their text. The cues are higher quantity, complexity, non-immediacy, expressiveness, informality, and affect; and less diversity, and specificity of language in their messages. Methods based on Information Retrieval from web were also proposed to verify authenticity of news articles. Banko et al. (2007) in their work extracted claims from web to match with that of a given document to find inconsistencies. To deal with the problem further, researchers have also tried to seek deep learning strategies in their work. Bajaj (2017) in his work applied various deep learning strategies on dataset composed of fake news articles available in Kaggle[2] and authentic news articles extracted from Signal Media News[3] dataset and observed that classifiers based on Gated Recurrent Unit (GRU), Long Short Term Memory (LSTM), Bi-directional Long Short Term Memory (Bi-LSTM) performed better than the classifiers based on CNN. Ma et al. (2016) in their work, focused on developing a system to detect Rumor at EVENT level rather than at individual post level. The approach was to look at a set of relevant posts to a event at a given time interval to predict veracity of the event. They showed that use of recurrent networks are particularly useful in this task. Dataset from two different social media platform, Twitter, and Weibo were used. Chen et al. (2017) further built on the work of Ma et al. (2016) for early detection Rumors at Event level, using the same dataset. They showed that the use of attention mechanism in recurrent network improves the performance in terms of precision, and recall measure, outperforming every other existing model for detecting rumor at an early stage. Castillo et al. (2011) used social media dataset (which is also used by Ma et al. (2016) for Rumor Detection) and developed a hybrid deep learning model which showed promising performance on both Twitter data and Weibo data. They showed that both, capturing the temporal behavior of the articles as well as learning source characteristics about the behavior of the users, are essential for

[2]https://www.kaggle.com/mrisdal/fake-news

[3]http://research.signalmedia.co/newsir16/signal-dataset.html

Table 1: Example statement of each class.

Ex	Statement (St)	Label
1	McCain opposed a requirement that the government buy American-made motorcycles. And he said all buy-American provisions were quote 'disgraceful.'	**T**
2	Almost 100,000 people left Puerto Rico last year.	**MT**
3	Rick Perry has never lost an election and remains the only person to have won the Texas governorship three times in landslide elections.	**HT**
4	Mitt Romney wants to get rid of Planned Parenthood.	**BT**
5	I dont know who (Jonathan Gruber) is.	**F**
6	Transgender individuals in the U.S. have a 1-in-12 chance of being murdered.	**PF**

Table 2: Meta-data related to each example. P, F, B, H, M is speaker's previous count of Pants-fire, False, Barely-true, Half-true, Mostly-true respectively.

Ex	St Type	Spk	Spk's Job	State	Party	P	F	B	H	M	Context
1	federal-budget	barack-obama	President	Illinois	democrat	70	71	160	163	9	a radio ad
2	bankruptcy, economy, population	jack-lew	Treasury secretary	Washington, D.C.	democrat	0	1	0	1	0	an interview with Bloomberg News
3	candidates-biography	ted-nugent	musician	Texas	republican	0	0	2	0	2	an oped column.
4	abortion, federal-budget, health-care	planned-parenthood -action-fund	Advocacy group	Washington, D.C.	none	1	0	0	0	0	a radio ad
5	health-care	nancy-pelosi	House Minority Leader	California	democrat	3	7	11	2	3	a news conference
6	corrections-and-updates, crime, criminal -justice, sexuality	garnet-coleman	president, ceo of Apartments for America, Inc.	Texas	democrat	1	0	1	0	1	a committee hearing

fake news detection. Further integrating these two elements improves the performance of the classifier.

Problems related to these topics have mostly been viewed concerning binary classification. Likewise, most of the published works also has viewed fake news detection as a binary classification problem (i.e., fake or true). But by observing very closely it can be seen that fake news articles can be classified into multiple classes depending on the fakeness of the news. For instance, there can be certain exaggerated or misleading information attached to a true statement or news. Thus, the entire news or statement can neither be accepted as completely true nor can be discarded as entirely false. This problem was addressed by Wang (2017) where they introduced **Liar** dataset comprising of a substantial volume of short political statements having six different class annotations determining the amount of fake content of each statement. In his work, he showed comparative studies of several statistical and deep learning based models for the classification task and found that the CNN model performed best. Long et al. (2017) in their work used the **Liar** dataset, and proposed a hybrid attention-based LSTM model for this task, which outperformed W.Yang's hybrid CNN model, establishing a new state-of-the-art.

In our current work we propose an ensemble architecture based on CNN (Kim, 2014) and Bi-

LSTM (Hochreiter and Schmidhuber, 1997), and this has been evaluated on **Liar** (Wang, 2017) dataset. Our proposed model tries to capture the pattern of information from the short statements and learn the characteristic behavior of the source speaker from the different attributes provided in the dataset, and finally integrate all the knowledge learned to produce fine-grained multi-class classification.

2 Methodology

We propose a deep multi-label classifier for classifying a statement into six fine-grained classes of fake news. Our approach is based on an ensemble model that makes use of Convolutional Neural Network (CNN) (Kim, 2014) and Bi-directional Long Short Term Memory (Bi-LSTM) (Hochreiter and Schmidhuber, 1997). The information presented in a statement is essentially sequential in nature. In order to capture such sequential information we use Bi-LSTM architecture. Bi-LSTM is known to capture information in both the directions: forward and backward. Identifying good features manually to separate true from fake even for binary classification, is itself, a technically complex task as human expert even finds it difficult to differentiate true from the fake news. Convolutional Neural Network (CNN) is known to capture the hidden features efficiently. We hypothesize that CNN will be able to detect hidden features of the given statement and the information related to the statements to eventually judge the authenticity of each statement. We make an intuition that both- capturing temporal sequence and identifying hidden features, will be necessary to solve the problem. As described in data section, each short statement is associated with 11 attributes that depict different information regarding the speaker and the statement. After our thorough study we identify the following relationship pairs among the various attributes which contribute towards labeling of the given statements.

Relation between: *Statement* and *Statement type*, *Statement* and *Context*, *Speaker* and *Party*, *Party* and *Speaker's job*, *Statement type* and *Context*, *Statement* and *State*, *Statement* and *Party*, *State* and *Party*, *Context* and *Party*, *Context* and *Speaker*.

To ensure that deep networks capture these relations we propose to feed each of the two attributes, say A_x and A_y, of a relationship pair into a separate individual model say M_i and M_j respectively. Then, concatenate the output of M_i and M_j and pass it through a fully connected layer to form an individual relationship network layer say $Network_n$ representing a relation. Fig. 1 illustrates an individual relationship network layer. Eventually after capturing all the relations we group them together along with the five-column attributes containing information regarding speaker's total credit history count. In addition to that, we also feed in a special feature vector that is proposed by us and is to be formed using the count history information. This vector is a five-digit number signifying the five count history columns, with only one of the digit being set to '1' (depending on which column has the highest count) and the rest of the four digits are set to '0'. The deep ensemble architecture is depicted in Fig. 2.

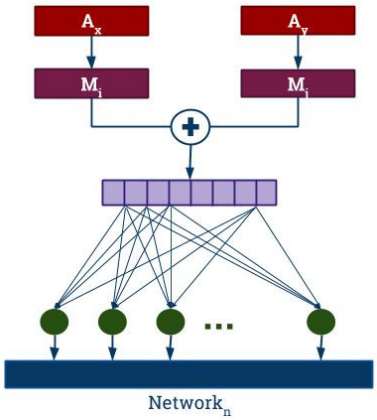


Figure 1: A relationship network layer. A_x and A_y are two attributes, M_i and M_j are two individual models, $Network_n$ is a representation of a network capturing a relationship

2.1 Bi-LSTM

Bidirectional LSTMs are the networks with LSTM units that process word sequences in both the directions (i.e. from left to right as well as from right to left). In our model we consider the maximum input length of each statement to be 50 (average length of statements is 17 and the maximum length is 66, and only 15 instances of the training data of length greater than 50) with post padding by zeros. For attributes like statement type, speaker's job, context we consider the maximum length of the input sequence to be 5, 20, 25, respectively. Each

input sequence is embedded into 300-dimensional vectors using pre-trained Google News vectors (Mikolov et al., 2013) (Google News Vectors 300dim is also used by Wang (2017) for embedding). Each of the embedded inputs are then fed into separate Bi-LSTM networks, each having 50 neural units at each direction. The output of each of these Bi-LSTM network is then passed into a dense network of 128 neurons with activation function as 'ReLU'.

2.2 CNN

Over the last few years many experimenters has shown that the convolution and pooling functions of CNN can be successfully used to find out hidden features of not only images but also texts. A convolution layer of $n \times m$ kernel size will be used (where m-size of word embedding) to look at n-grams of words at a time and then a MaxPooling layer will select the largest from the convoluted inputs.The attributes, namely speaker, party, state are embedded using pre-trained 300-dimensional Google News Vectors (Mikolov et al., 2013) and then the embedded inputs are fed into separate Conv layers.The different credit history counts the fake statements of a speaker and a feature proposed by us formed using the credit history counts are directly passed into separate Conv layers.

2.3 Combined CNN and Bi-LSTM Model

The representations obtained from CNN and Bi-LSTM are combined together to obtain better performance.

The individual dense networks following the Bi-LSTM networks carrying information about the statement, the speaker's job, context are reshaped and then passed into different Conv layers. Each convolution layer is followed by a Maxpooling layer, which is then flattened and passed into separate dense layers. Each of the dense layers of different networks carrying different attribute information are merged, two at a time-to capture the relations among the various attributes as mentioned at the beginning of section 2. Finally, all the individual networks are merged together and are passed through a dense layer of six neurons with softmax as activation function as depicted in. The classifier is optimized using Adadelta as optimization technique with categorical cross-entropy as the loss function.

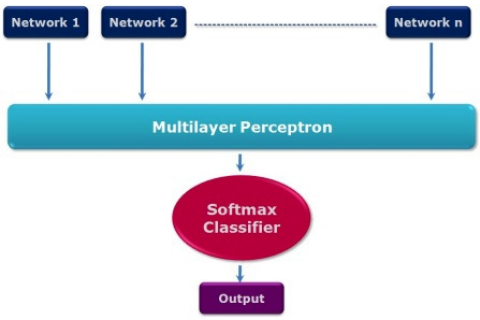


Figure 2: Deep Ensemble architecture

3 Data

We use the dataset, named **LIAR** (Wang, 2017), for our experiments. The dataset is annotated with six fine-grained classes and comprises of about 12.8K annotated short statements along with various information about the speaker. The statements which were mostly reported during the time interval [2007 to 2016], are considered for labeling by the editors of **Politifact.com**. Each row of the data contains a short statement, a label of the statement and 11 other columns correspond to various information about the speaker of the statement. Descriptions of these attributes are given below:

1. **Label**: Each row of data is classified into six different types, namely
   (a) **Pants-fire (PF):** Means the speaker has delivered a blatant lie .
   (b) **False (F):** Means the speaker has given totally false information.
   (c) **Barely-true (BT):** Chances of the statement depending on the context is hardly true. Most of the contents in the statements are false.
   (d) **Half-true (HT):** Chances of the content in the statement is approximately half.
   (e) **Mostly-true (MT):** Most of the contents in the statement are true.
   (f) **True (T):** Content is true.
2. **Statement by the politician:** This statement is a short statement.
3. **Subjects:** This corresponds to the content of the text. For examples, foreign policy, education, elections etc.
4. **Speaker:** This contains the name of the speaker of the statement.

5. **Speaker's job title:** This specifies the position of the speaker in the party.
6. **State information:** This specifies in which state the statement was delivered.
7. **Party affiliation:** This denotes the name of the party of the speaker belongs to.
8. The next five columns are the counts of the speaker's statement history. They are:
   (a) **Pants fire count;**
   (b) **False count;**
   (c) **Barely true count;**
   (d) **Half false count;**
   (e) **Mostly true count.**
9. **Context:** This corresponds to the venue or location of the speech or statement.

The dataset consists of three sets, namely a training set of 10,269 statements, a validation set of 1,284 statements and a test set of 1,266 statements.

4 Experiments and Results

In this section, we report on the experimental setup, evaluation results, and the necessary analysis.

4.1 Experimental Setup

All the experiments are conducted in a python environment. The libraries of python are required for carrying out the experiments are **Keras**, **NLTK**, **Numpy**, **Pandas**, **Sklearn**. We evaluate the performance of the system in terms of accuracy, precision, recall, and F-score metrics.

4.2 Results and Analysis

We report the evaluation results in Table 3 that also show the comparison with the system as proposed by Wang (2017) and Long et al. (2017).

Table 3: Overall evaluation results

Model	Network	Attributes taken	Accuracy
William Yang Wang (2017)	Hybrid CNN	All	0.274
Y. Long (2017)	Hybrid LSTM	All	0.415
Bi-LSTM Model	Bi-LSTM	All	0.4265
CNN Model	CNN	All	0.4289
Our Proposed Model	RNN-CNN combined	All	**0.4487**

We depict the overall evaluation results in Table 3 along with the other existing models. This shows that our model performs better than the existing state-of-the-art model as proposed in Long et al. (2017). This state-of-the-art model was a hybrid LSTM, with an accuracy of 0.415. On the other hand, our proposed model shows 0.4265, 0.4289 and 0.4487 accuracies for Bi-LSTM, CNN and the combined CNN+Bi-LSTM model, respectively. This clearly supports our assumption that capturing temporal patterns using Bi-LSTM and hidden features using CNN are useful, channelizing each profile attribute through a different neural layer is important, and the meaningful combination of these separate attribute layers to capture relations between attributes, is effective.

Table 4: Evaluation of our different proposed deep learning models on basis of precision, recall, and F1 score. PF, F, BT, HT, MT, and T are class *pants-fire, fale, barely-true, half-true, mostly-true, and true respectively.*

Bi-LSTM model				
	precision	**recall**	**F1-score**	**Support**
PF	0.73	0.35	0.47	92
F	0.47	0.53	0.50	249
BT	0.58	0.32	0.41	212
HT	0.39	0.46	0.42	265
MT	0.33	0.66	0.44	241
T	0.88	0.14	0.23	207
Avg/Total	0.53	0.43	0.41	1266
CNN model				
PF	0.67	0.39	0.49	92
F	0.36	0.63	0.46	249
BT	0.50	0.36	0.42	212
HT	0.42	0.46	0.44	265
MT	0.41	0.49	0.45	241
T	0.70	0.16	0.26	207
Avg/Total	0.48	0.43	0.42	1266
Combined model				
PF	0.70	0.43	0.54	92
F	0.45	0.61	0.52	249
BT	0.61	0.32	0.42	212
HT	0.35	0.73	0.47	265
MT	0.50	0.36	0.42	241
T	0.85	0.14	0.24	207
Avg/Total	**0.55**	**0.45**	**0.43**	1266

We also report the precision, recall and F-score measures for all the models. Table 4 depicts the evaluation results on the test data of our proposed CNN, Bi-LSTM and CNN and Bi-LSTM combined models. The evaluation shows that on the precision measure the combined model performs best with an average precision of **0.55** while that of Bi-LSTM model is 0.53 and CNN model is 0.48. The combined model of CNN and Bi-LSTM even performs better with respect to recall and F1-Score measures. The combined model yields the average recall of **0.45** and average F1-score

Table 5: Confusion matrix of our different proposed models on Test data. PF, F, BT, HT, MT, and T are class *pants-fire, fale, barely-true, half-true, mostly-true, and true respectively.*

Bi-LSTM model						
Actual\Predicted	**PF**	**F**	**BT**	**HT**	**MT**	**T**
PF	32	35	3	8	14	0
F	4	131	16	36	59	3
BT	5	31	68	48	60	0
HT	0	38	8	123	95	1
MT	1	20	8	54	158	0
T	2	25	15	47	90	28
CNN model						
PF	36	35	6	11	2	2
F	7	156	21	30	28	7
BT	5	66	76	34	29	2
HT	2	75	14	123	48	3
MT	1	53	17	51	119	0
T	3	44	18	44	65	33
Combined model						
PF	40	34	4	10	4	0
F	7	152	10	67	11	2
BT	4	48	68	83	9	0
HT	0	43	7	193	20	2
MT	2	31	9	112	86	1
T	4	31	13	89	41	29

of **0.43** while that of Bi-LSTM model is 0.43 and 0.41, respectively and of the CNN model is 0.43 and 0.42, respectively. On further analysis, we observe that although the performance (based on precision, recall, and F1-score) of each of the models for every individual class is close to the average performance, but in case of the prediction of the class label **TRUE** the performance of each model varies a lot from the respective average value. The precisions of TRUE is promising (Bi-LSTM model:0.88, CNN model: 0.7, Combined model:**0.85**), but the recall (Bi-LSTM model:0.14, CNN model: 0.16, Combined model:0.14) and the F1-score (Bi-LSTM model:0.23, CNN model: 0.26, Combined model:0.24) are very poor. This entails the fact that our proposed model predicts comparatively less number of instances as TRUE, but when it does the prediction is very accurate. Thus it can be claimed that if a statement is predicted as **True** by our proposed model then one can rely on that with high confidence. Although our model performs superior compared to the existing state-of-the-art, still the results were not error free. We closely analyze the models' outputs to understand their behavior and perform both quantitative as well as qualitative error analysis. For quantitative analysis, we create the confusion matrix for each of our models. Confusion matrix corresponding to the experiment with proposed Bi-LSTM model, corresponding to experiment with proposed CNN model, and corresponding to our final experiment i.e with proposed RNN-CNN combined model is given in Table 5.

From these quantitative analysis it is seen that in majority of the cases the test data statements originally labeled with **Pants-Fire** class gets confused with the **False** class, statements originally labeled as **False** gets confused with **Barely true** and **half true** classes, statements originally labeled as **Half true** gets confused with **Mostly True** and **False** class, statements originally labeled as **Mostly true** gets confused with **Half True**, statements originally labeled with **True** gets confused with **Mostly True** class.

It is quite clear that errors were mostly concerned with the classes, overlapping in nature. Confusion is caused as the contents of the statements belonging to these classes are quite similar. For example, the difference between 'Pants-Fire' and 'False' class is that only the former class corresponds to the false information with more intensity. Likewise 'Half True' has high similarity to 'False', and 'True' with 'Mostly True'. The difference between 'True' and 'Mostly True' is that the later class has some marginal amount of false information, while the former does not.

For qualitative analysis, we closely look at the actual statements and try to understand the causes of misclassifications. We come up with some interesting facts. There are some speakers whose statements are not present in the training set, but are present in the test set. For few of these statements, our model tends to produce wrong answers. Let us consider the example given in Table 6. For this speaker, there is no training data available and also the count history of the speaker is very less. So our models assign an incorrect class. But it is to be noted that even if there is no information about the speaker in the training data and the count history of the speaker is almost empty, still we are able to generate a prediction of a class that is close to the original class in terms of meaning.

It is also true that classifiers often make mistakes in making the fine distinction between the classes due to the insufficient number of training instances. Thus, classifiers tend to misclassify the

Table 6: Sample text with wrongly predicted label and original label. Spk is speaker, and P, F, B, H, M is speaker's previous count of Pants-fire, False, Barely-true, Half-true, Mostly-true respectively.

Label	Statement	St Type	Spk	Spk's Job	State	Party	Context	P	F	B	H	M	Predicted Label
barely-true	We know there are more Democrats in Georgia than Republicans. We know that for a fact.	elections	mike-berlon	none	Georgia	democrat	an article	1	0	0	0	0	False

instances into one of the nearby (and overlapped) classes.

5 Conclusion and Future Works

In this paper, we have tried to address the problem of fake News detection by looking into short political statements made by the speakers in different types of daily access media. The task was to classify any statement into one of the fine-grained classes of fakeness. We have built several deep learning models, based on CNN, Bi-LSTM and the combined CNN and Bi-LSTM model. Our proposed approaches mainly differ from previously mentioned models in system architecture, and each model performs better than the state of the art as proposed by Long et al. (2017), where the statements were passed through one LSTM and all the other details about speaker's profile through another LSTM. On the other hand, we have passed every different attribute of speaker's profile through a different layer, captured the relations between the different pairs of attributes by concatenating them. Thus, producing a meaningful vector representation of relations between speaker's attributes, with the help of which we obtain the overall accuracy of 44.87%. By further exploring the confusion matrices we found out that classes which are closely related in terms of meaning are getting overlapped during prediction. We have made a thorough analysis of the actual statements, and derive some interesting facts. There are some speakers whose statements are not present in the training set but present in the test set. For some of those statements, our model tends to produce the wrong answers. This shows the importance of speakers' profile information for the task. Also as the classes and the meaning of the classes are very near, they tend to overlap due to less number of examples in training data.

We would like like to highlight some of the possible solutions to solve the problems that we encountered while attempted to solve fake news detection problem in a more fine-grained way.

- More labeled data sets are needed to train the model more accurately. Some semi-supervised or active learning models might be useful for this task.
- Along with the information of a speaker's count history of lies, the actual statements are also needed in order to get a better understanding of the patterns of the speaker's behavior while making a statement.

Fake news detection into finely grained classes that too from short statements is a challenging but interesting and practical problem. Hypothetically the problem can be related to **Sarcasm detection** (Joshi et al., 2017) problem. Thus it will also be interesting to see the effect of implementing the existing methods that are effective in sarcasm detection domain in Fake News detection domain.

References

Samir Bajaj. 2017. the pope has a new baby ! fake news detection using deep learning.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2670–2676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Alessandro Bessi and Emilio Ferrara. 2016. Social bots distort the 2016 u.s. presidential election online discussion. *First Monday*, 21(11).

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 675–684, New York, NY, USA. ACM.

Tong Chen, Lin Wu, Xue Li, Jun Zhang, Hongzhi Yin, and Yang Wang. 2017. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection. *CoRR*, abs/1704.05973.

Venkatesh Duppada. 2018. "attention" for detecting unreliable news in the information age. In *AAAI Workshops*.

N. Eshraqi, M. Jalali, and M. H. Moattar. 2015. Spam detection in social networks: A review. In *2015 International Congress on Technology, Communication and Knowledge (ICTCK)*, pages 148–152.

Miriam Fernandez and Harith Alani. 2018. Online misinformation: Challenges and future directions. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 595–602, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Aditi Gupta, Ponnurangam Kumaraguru, Carlos Castillo, and Patrick Meier. 2014. Tweetcred: Real-time credibility assessment of content on twitter. In *International Conference on Social Informatics*, pages 228–243. Springer.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Aditya Joshi, Pushpak Bhattacharyya, and Mark J. Carman. 2017. Automatic sarcasm detection: A survey. *ACM Comput. Surv.*, 50(5):73:1–73:22.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. 2017. Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 252–256. Asian Federation of Natural Language Processing.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 3818–3824. AAAI Press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Victoria Rubin, Nadia Conroy, and Yimin Chen. 2015. Towards news verification: Deception detection methods for news discourse.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36.

Eugenio Tacchini, Gabriele Ballarin, Marco L. Della Vedova, Stefano Moret, and Luca de Alfaro. 2017. Some like it hoax: Automated fake news detection in social networks. *CoRR*, abs/1704.07506.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426. Association for Computational Linguistics.

L. Zhou, D. P. Twitchell, Tiantian Qin, J. K. Burgoon, and J. F. Nunamaker. 2003. An exploratory study into deception detection in text-based computer-mediated communication. In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, pages 10 pp.–.

Building Discourse Parser for Thirukkural

R. Anita, C.N. Subalalitha
Department of Computer Science and Engineering
SRM Institute of Science and Technology
Kattankulathur-603 203, Tamilnadu, India
{anitaramalingam17,subalalitha}@gmail.com

Abstract

Thirukkural is one of the famous Tamil Literatures in the world. It was written by Thiruvalluvar, and focuses on ethics and morality. It provides all possible solutions to lead a successful and a peaceful life fitting any generation. It has been translated into 82 global languages, which necessitate the access of Thirukkural in any language on the World Wide Web (WWW) and processing the Thirukkural computationally. This paper aims at constructing the Thirukkural Discourse Parser which finds the semantic relations in the Thirukkurals which can extract the hidden meaning in it and help in utilizing the same in various Natural Language Processing (NLP) applications, such as, Summary Generation Systems, Information Retrieval (IR) Systems and Question Answering (QA) Systems. Rhetorical Structure Theory (RST) is one of the discourse theories, which is used in NLP to find the coherence between texts. This paper finds the relation within the Thriukkurals and the discourse structure is created using the Thirukkural Discourse Parser. The resultant discourse structure of Thirukkural can be indexed and further be used by Summary Generation Systems, IR Systems and QA Systems. This facilitates the end user to access Thirukkural on WWW and get benefited. This Thirukkural Discourse Parser has been tested with all 1330 Thirukurals using precision and recall.

1 Introduction

Tamil literature has so many nuggets hidden in it which need to be explored for the goodness of the society. One of the ways to explore the Tamil literature is to make it easily accessible on the World Wide Web (WWW). For instance, Thirukkural is one of the famous Tamil literatures in the world and it is respected by people across the globe. In order to make it to reach to all people, it should be made available on the web. This makes necessary to process it computationally. Hence, this paper proposes a methodology to perform a discourse analysis of Thirukkural, which aids in exploring its semantics and also organizing it on the web.

Natural Language Processing (NLP) is the process of interaction between computer and human or natural languages. Analysis of text can be done at various levels namely, word, clause, sentence, paragraph and document. Discourse analysis is used for analyzing the text beyond the clause level. The proposed work attempts to extract the relations found within the Thirukkural.

Discourse structure of a text can be built by using a popular theory called, Rhetorical Structure Theory (RST) (Thompson and Mann, 1987; Mann and Thompson, 1988). Using RST-based discourse relations, the RST captures the coherence among the Natural Language (NL) text spans. The coherence can be found between two or more text fragments. The text fragments could be within a sentence, across sentences, across paragraphs and even across documents.

In this paper, each Thirukkural is considered as a sentence and discourse parser is built using RST. The contributions of this paper are twofold.

1) Finding feature set using rule based approach.
2) Building Discourse parser by identifying discourse relations.

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 18–25
Hyderabad, India, December 2019.

The rest of the paper is organized as follows. Section 2 describes background details. Section 3 describes the related work. Section 4 discusses the proposed work. Section 5 gives details on the evaluation of the proposed technique. Section 6 presents the conclusion and future works.

2 Background

This section describes about the Thirukkural and the basics of RST based Discourse Parsing.

2.1. Thirukkural

Thirukkural consists of 1330 couplets or Thirukkurals. They are classified into three sections and 133 chapters. Each chapter in Thirukkural has a specific subject and consists of ten couplets or Thirukkurals. A couplet consists of two lines. Each Thirukkural or couplet is formed with seven cirs (words). First line of the couplet consists of four cirs and the second line of the couplet consists of three cirs. A single Tamil word or a combination of two or more Tamil words forms a cir.

2.2. Rhetorical Structure Theory

RST is a descriptive theory which focuses on the organization of the natural language. It was proposed by Bill Mann, Sandy Thompson, and Christian Matthiessen at the University of Southern California (Thompson and Mann, 1987; Mann and Thompson, 1988). It identifies the coherence between the text spans using discourse relations and forms a discourse structure called rhetorical structure. The discourse units are Nucleus, Satellite and Discourse Relations. The nucleus carries the necessary information and the satellite carries the additional information supporting the nucleus.

Discourse relations are organized into three categories, namely, subject matter, presentational, and multinuclear. In subject matter relations, satellite is a request or problem posed by the reader, i.e. satisfied or solved by nucleus. Elaboration, evaluation and condition are some of the subject matter relations. In presentational relations, satellite increases reader's inclination in accepting the facts stated in nucleus. Antithesis, background and enablement are some of the presentational relations. In multinuclear relations, two nuclei are connected instead of one nucleus and one satellite. Conjunction, contrast and sequence are some of the multinuclear relations.

Figure 1 shows an example of how the nucleus, satellite, and the discourse relation are identified for an English sentence in Example 1.

Example 1 Raj sings well but he could not win the contest.

Nucleus:	Raj sings well
Satellite:	he could not win the contest
Discourse Relation:	Antithesis

Figure 1. NRS Sequence for Example 1.

In Example 1, the sentence holds antithesis relation. It is identified by the signal word *but*. "Raj sings well" is the nucleus, because it represents the ideas favored by the author. "He could not win the contest" is the satellite, because it represents the ideas disfavored by the author. These NRS sequences capture the inherent semantics in the texts which is applied to the Thirukkural couplets by the proposed approach.

3 Related Work

Subba and Di (2009) found discourse relations by using shift reduce parsing model and WordNet. The linguistic cues were used as features. The document was analyzed at sentence level. Hernault et al. (2010) constructed discourse parser by building discourse tree using Support Vector Machine Classifier. The document was analyzed beyond the sentence level and the combination of syntactic and lexical features such as words, POS tags and lexical heads were used as feature sets.

Hernault et al. (2010a) used a semi-supervised method called Feature Vector Extension for discourse relation classification. The method was based on the analysis of co-occurring features present in unlabelled data, which was then taken into account for extending the feature vectors given to a classifier. The word pairs, production rules from parse trees and Lexico-Syntactic context at the border between two units of text were used as features for the algorithm.

Sucheta et al. (2011) identified explicit discourse connectives for Penn Discourse Tree Bank (PDTB). They proposed shallow discourse parsing for performing token level argument segmentation. The document was analyzed at sentence level. The lexical, syntactic and semantic features were used as features.

Sucheta et al. (2012) improved a shallow discourse parser by using a constraint-based method based on conditional random fields and the recall was improved. Sucheta, Giuseppe, and Richard (2012) constructed a parser which uses local constraints and then global constraints. They analyzed the text at the inter sentence level and they used the lexico-syntactic features.

Subalalitha and Ranjani Parthasarathi (2013) used Tamil and Sanskrit literature concepts called suthras and sangatis, along with the current-day text processing theories namely, RST, Universal Networking Language for identifying semantic indices for Tamil documents. Suthras are used for representing the text in a crisp manner.

Sobha et al. (2014) and Sobha and Patnaik (2004) proposed automatic identification of connectives and their arguments for the Indian languages Hindi, Malayalam and Tamil. They used Conditional Random Fields machine learning technique. They used 3000 sentences from a health domain as a corpus. Sobha et al. (2014), annotated the three language corpus, namely Tamil Hindi and Malayalam, with the discourse relations.

Lin et al. (2014) constructed an end-to-end discourse parser in the PDTB style. Their parser identified all discourse and non-discourse relations, labeled the arguments, and found the sense of relation between arguments. The document was analyzed at paragraph level. The lexical, syntactic and semantic features were used as features.

Yangfeng and Jacob (2014) transformed surface features into a latent space by using a representation learning approach that facilitates RST discourse parsing. They used shift reduced discourse parser and analyzed the document at sentence level.

Uladzimir et al. (2015) segmented the German text for the RST-based discourse parsing. They analyzed the text at sentence level. Parminder et al. (2015) proposed document level sentiment analysis using RST discourse parsing and recursive neural network. They analyzed the text at document level and lexical features were used as features.

Subalalitha and Ranjani Parthasarathi (2015) found 13 RST Relations in Tamil documents. The Naïve Bayes probabilistic classifier machine learning algorithm was used and the Tamil documents were analyzed beyond the sentence level. The high level semantic features were used by their discourse parser, which were inherited from UNL to construct rhetorical structure trees.

Manfred et al. (2016) annotated the corpus with two theories, namely, RST and Segmented Discourse Representation Theory. It was also annotated with the argumentation annotation. The document was analyzed at sentence level. The syntactic and semantic features were used as feature set.

Yangfeng et al. (2016) proposed a latent variable recurrent neural network for finding the discourse relation between adjacent sentences. They analyzed the text at inter sentence level and they have used lexical features. Yangfeng and Noah (2017) proposed text categorization by using recursive neural network and RST. The document was analyzed at sentence level.

It can be observed that, the existing discourse methodologies analyzed the text in English documents and expository type Tamil documents. This paper proposes a discourse methodology that makes use of RST to identify the semantic relations/discourse relations from a Tamil literature text which lacks a regular pattern for semantic analysis. Unlike English which has a fixed SVO (Subject Verb Object) sentence pattern, Tamil expository texts have either SVO or SOV (Subject Object Verb) pattern. Tamil literatures on the other side neither follow SOV nor SVO pattern. Tamil literatures also have a relatively rich set of morphological variants (Anand et al., 2010; Goldsmith, 2001). This makes the processing of Tamil literature more complex than processing the expository Tamil documents. This paper focuses on finding discourse relations in a Tamil literary work called, Thirukkural, which has the structure of classic Tamil language poetry form, called venba. Venba style Tamil literature consists of lines between two and twelve. Expository Tamil documents have the cue words in middle of the sentence. It is not difficult to find the nucleus satellite identification for expository type of texts, whereas the cue words in Tamil literature specifically in venba style of texts will be present in any part of the sentence. If the cue word is present in the middle of the Thirukkural couplet, it is not difficult to find the nucleus satellite identification. If it is present at either end of the Thirukkural couplet then it is difficult to find the nucleus satellite identification. The proposed

Thirukkural Discourse Parser handles these cases to an extent which is discussed in the upcoming sections.

4 Proposed Work

The Tamil Thirukkural couplets are given as input. Initially the cue phrases or signal words are identified in each Thirukkural. RST based discourse relations are identified by Thirukkural Discourse Parser based on the cue words and semantics. Then the Nucleus and Satellites are identified for each Thirukkural. Finally, the NRS sequences are identified as output from the Thirukkural Discourse Parser.

4.1. Feature Sets

The connectives connecting two clauses of the Thirukkural are used as the feature set as they signal a discourse relation. An analysis of the 600 Thirukkural has been done and the feature set for each discourse relation has been identified. For condition relation, 108 features have been identified; for evidence relation, 36 features have been identified; for contrast relation, 37 features have been identified; for enablement relation, 24 features have been identified; for background relation, 9 features have been identified. The feature sets, cue words and signal words are interchangeably used in this paper. The part of the cue words are appeared in Table 1. For example, ‘இலவே (Ilave-If not)’, ‘ஆயின் (Ayin-If)’, and ‘ஆற்றின் (Arrin-If someone did)’ are some of the cue words commonly appeared in Thirukkural.

4.2. Discourse Relation Identification

The discourse relations namely, condition, evidence, contrast, enablement and background are identified by the Thirukkural Discourse Parser. A cue word may either be a single word which can be explicitly identified by the Thirukkural Discourse Parser or it may be a case suffix which may have to be split by the morphological analyzer (Anandan et al., 2001).

If the cue words explicitly appear in the Thirukkural, then the RST based discourse relations are identified using the signal words in Table 1. The cue words are given in Tamil along with their English transliteration and English meaning.

If the cue words do not explicitly appear in the Thirukkural, then the morphological analyzer is used for finding the cue words. For example, in the word ‘எழுத்தெல்லாம் (Eluttellam-All the letters) ’, the cue word ‘எல்லாம் (Ellam-Everything)’ is a case suffix and so the morphological analyzer is used to isolate the cue word (‘எழுத்து+ எல்லாம்’). Now the cue word ‘எல்லாம் (Ellam- Everything)’ can be used for identifying the RST based discourse relation.

S. No.	Relation	List of Cue words
1	Condition	இலவே (Ilave-If not), என்னும் (Ennum- The), பெறின் (Perin- If received), என்னாம் (Ennam-If), ஆயின் (Ayin-If), ஆற்றின் (Arrin-If someone did)
2	Evidence	வேண்டா (Venta- Do Not), அதுவல்லது (Atuvallatu - That is not), தான்(Tan- Just), போன்று (Ponru- Like), தேரின் (Terin- Selection), எங்ஙனம் (Ennanam- How)
3	Contrast	அரிய (Ariya – Rare), மற்றெல்லாம் (Marrellam- On every), ஆதல் (Atal- Therefore), உய்க்கும் (Uykkum- That derived)
4	Enablement	எல்லாம் (Ellam- Everything), அவருள்ளும் (Avarullum- Plunge), மன்ற (Manra- House), போல(Pola-Like)
5	Background	எனினும் (Eninum- However), செல்லாது (Cellatu- Invalid), இனிதே (Inite- Greeter), அற்று (Arru- Without), உறையும் (Uraiyum- Freezing)

Table 1. Some Relations and Cue Words

4.3. Nuclearity Identification

The cue words are appeared anywhere in the Thirukkural. In most of the Thirukkural, it is appeared in the middle. The text before the cue word is considered as Clause1 and the text after the cue word is considered as Clause2. Clause1 and Clause2 can be indicated as nucleus and satellite.

In few Thirukkural, Clause1 can act as the nucleus and Clause2 can act as the satellite. And in others, they may be vice versa. Hence, it is identified separately and the NRS sequences are identified accordingly. Table 2 lists some of the cue words appear in Thirukkural in which Clause1 and Clause2 are categorized as nucleus and satellite.

Example 2 in Figure 2 shows NRS sequence identified by the Thirukkural Discourse Parser.

S. No	Relation	Clause1-Nucleus, Clause2-Satellite	Clause2-Nucleus, Clause1-Satellite
1	Condition	இலவே (Ilave-If not), என்னும் (Ennum-The), பெறின் (Perin- If received), என்னாம் (Ennam-If), வைப்பின் (Vaippin-Fund)	ஆயின் (Ayin-If), ஆற்றின் (Arrin-If someone did)
2	Evidence	வேண்டா (Veṇṭa- Do Not), அதுவல்லது (Atuvallatu - That is not),	தான் (Tan- Just), போன்று (Ponru- Like), தேரின் (Terin- Selection), எங்ஙனம் (Ennanam- How)
3	Contrast	அரிய (Ariya – Rare), மற்றெல்லாம் (Marrellam- On every), ஆதல் (Atal- Therefore), உய்க்கும் (Uykkum- That derived)	உரியர் (Uriyar- Belong) , செயினும் (Ceyinum- Though did)
4	Enablement	எல்லாம் (Ellam- Everything), அவருள்ளும் (Avarullum- Plunge), மன்ற (Manra- House)	போல (Pola-Like)
5	Background	செல்லாது (Cellatu- Invalid), இனிதே (Inite-Greeter), அற்று (Arru- Without), உறையும் (Uraiyum- Freezing)	எனினும் (Eninum- However),

Table 2. Nucleus and Satellite Identification

The Thirukkural in Figure 2 is given as the input for the Thirukkural Discourse Parser. This Thirukkural has the cue word 'போல (Pola-Like)' explicitly. This cue word is used for identifying the Enablement relation. The Thirukkural Discourse Parser identifies 'Enablement' as the discourse relation, 'ஆங்கே இடுக்கண் களைவதாம் நட்பு' as nucleus, (as it contains an action), and 'உடுக்கை இழந்தவன் கை' as satellite, (as it contains the information for performing the action). Similarly NRS sequences for all such cases present in the Thirukkurals are identified by using the Thirukkural Discourse Parser. Figure 3 shows Nucleus, Satellite and Discourse relation for the Example 2.

Example 2:
உடுக்கை இழந்தவன் கைபோல ஆங்கே இடுக்கண் களைவதாம் நட்பு.

English Transliteration:
Utukkai ilantavan kaipola anke itukkan kalaivatam natpu.

Meaning in English:
True friendship hastens to the rescue of the afflicted as readily as the hand of one whose garment is loosened.

Figure 2. Example 2

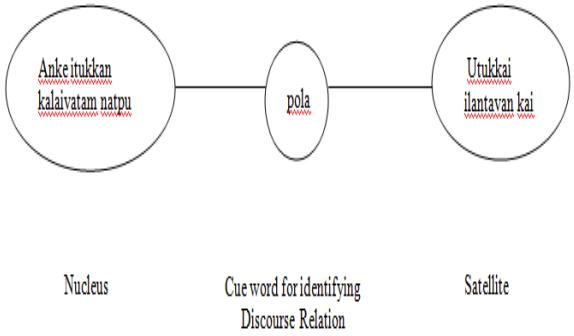


Figure 3. NRS Sequence for Example 2.

The algorithm for Thirukkural Discourse Parser is shown in Figure 4.

Input: Thirukkural Couplets
Output: NRS Sequences
(i) Find cue words in all Thirukkural Couplets
(ii) Store it separately corresponding to the relation
(iii) **for** Thirukkural couplets = 1 to 1330 **do**
if cue word is present in the Thirukkural **then**

identify the Discourse relation
display the NRS sequences
end
if cue word is not present in the Thirukkural **then**
use morphological analyzer to find the cue word

identify the Discourse relation
display the NRS sequences
end
end

Figure 4. Algorithm for Thirukkural Discourse Parser

5 Evaluation

The features are extracted from 600 couplets. The 1330 Thirukkural couplets are given as the input for the Thirukkural Discourse Parser. The NRS sequences are identified by the Thirukkural Discourse Parser based on the cue words. This work is evaluated using the parameters, precision and recall. Precision (P), and recall (R) values are calculated using equations (1) and (2).

$$\text{Precision(P)} = \frac{\text{Number of relevant NRS sequences retrieved, (C)}}{\text{Total number of NRS sequences retrieved,(M)}} \quad (1)$$

$$\text{Recall(R)} = \frac{\text{Number of relevant NRS sequences retrieved, (C)}}{\text{Total number of relevant NRS sequences present, (N)}} \quad (2)$$

The Table 3 shows the precision and recall of the discourse parser. The total number of NRS sequences retrieved by the proposed Thirukkural Discourse Parser is denoted as - M, total number of relevant NRS sequences actually present in Thirukkural is denoted as - N, and number of relevant NRS sequences retrieved by the proposed Thirukkural Discourse Parser is denoted as - C. The value of the variables, C and N are calculated using human judgement. About six domain experts have calculated these metric N, and the average has been taken and presented in Table 3.

Relation	N	M	C	P=$\frac{C}{M}$ (%)	R=$\frac{C}{N}$ (%)
Condition	547	616	514	83.44	93.97
Evidence	248	283	225	79.51	90.73
Contrast	262	216	182	84.26	75.21
Enablement	189	158	129	81.65	76.33
Background	176	145	114	78.62	73.08

Table 3. Precision, Recall and F-Measure for the Discourse Relation

It can be observed from the table that the total number of NRS sequences relevant to the condition relation is 547 and total number of NRS sequences retrieved from the Thirukkural Discourse Parser is 616. This is because more than one cue words belonging to the Condition relation are appeared in some Thirukkurals.

In Example 3 shown in Figure 5, two cue words "*ஆற்றின்* (Arrin-If someone did)" and "*பெறின்* (Perin- If received)" have appeared in the Thirukkural. Both cue words belong to the Condition relation. In order to analyze the significance of each feature, the NRS sequences emerging out of two features present in the same Thirukkural is counted and hence, M is greater than N. Similarly, for Evidence relation also M is greater than N. On the other side, in Contrast, Enablement and Background relations, N is more than M.

Example 3:
செறிவறிந்து சீர்மை பயக்கும் அறிவறிந்து ஆற்றின் அடங்கப் பெறின்.

English Transliteration:
Cerivarintu cirmai payakkum arivarintu aarrin atankap perin.

Meaning in English:
Knowing that self-control is knowledge, if a man should control himself, in the prescribed course, such self-control will bring him distinction among the wise.

Figure 5. Example 3.

The correctly retrieved Thirukkurals, C is smaller than N in all the discourse relation, this is due to the inability of the discourse parser to extract the NRS sequences using implicit cue words.

Example 4:
உள்ளற்க உள்ளம் சிறுகுவ கொள்ளற்க அல்லற்கண் ஆற்றறுப்பார் நட்பு.

English Transliteration:
Ullarka ullam cirukuva kollarka allarkaṇ arraruppar natpu.

Meaning in English:
Do not think of things that discourage your mind, nor contract friendship with those who would forsake you in adversity.

Figure 6. Example 4.

In Example 4 shown in Figure 6, the cue word "போல (Pola-Like)" is implicit. The Enablement relation identification needs additional semantic analysis which is currently not done by the discourse parser.

In some Thirukkurals, more than one cue words pointing to different relations have appeared. Therefore more than one NRS sequences are identified by the Thirukkural Discourse Parser for the same Thirukkural.

In Example 5 shown in Figure 7, two cue words namely, "ஆற்றின் (Arrin-If someone did)" and "போல் (Pol-Like)" are present.

"ஆற்றின் (Arrin-If someone did)" is a cue word related to Condition relation and "போல் (Pol-Like)" is a cue word related to Evidence relation. So the Condition and Evidence relations are identified by the Thirukkural Discourse Parser.

Example 5:
ஒருமையுள் ஆமைபோல் ஐந்தடக்கல் ஆற்றின்
எழுநம்யும் ஏமாப் புடைத்து.

English Transliteration:
Orumaiyul amaipol aintatakkal arrin
elunamyum emap putaittu.

Meaning in English:
Should one throughout a single birth, like a tortoise keep in his five senses, the fruit of it will prove a safe-guard to him throughout the seven-fold births.

Figure 7. Example 5.

The precision and recall values of the discourse parser can further be increased by increasing the feature sets, by incorporating a machine learning algorithm. The efficiency can be increased by finding the discourse relations for the Thirukkurals having implicit cue words. The efficiency of the Thirukkural Discourse Parser also depends on the efficiency of the morphological analyzer. A high level semantic knowledge base such as WordNet (George, 1995) or ontology may improve the efficiency even better.

6 Conclusion and Future Works

Thirukkural has much valuable information that is to be followed by the society. In order to access the Thirukkural on the web, the semantic analysis of the same becomes necessary. This paper makes use of discourse theory, named, RST to do a discourse/semantic analysis on the Thirukkural which will be useful to retrieve the Thirukkural using an Information Retrieval System.

Keyword-based Thirukkural search is available but limits the user to retreive the Thirukkural containing only the query words. In this paper, we propose a methodology to construct a discourse parser for Thirukkural which aids in semantic analysis and semantic representation of Thirukkural. This kind of semantic representation can be used for efficient semantic indexing of Thirukkural for better retrieval.

Using the results of the proposed discourse parser, an analysis of how Thirukkural is written and organized is also evident which can be useful to write similar works in future. This kind of analysis can also help in automatic author detection of text (Stamatatos, 2008).

This paper focuses on the discourse relations present within a Thirukkural couplet. It may be further extended to find the relations across the Thirukkural couplets. An application that clearly depicts the discourse structure representation is also to be done.

References

Anandan, P., Ranjani Parthasarathy, and Geetha, T.V. 2001. Morphological Analyzer for Tamil. *ICON 2002*, RCILTS-Tamil, Anna University, India.

Anand Kumar, M., Dhanalakshmi, V., Soman, K.P., and Rajendran, S. 2010. A Sequence Labeling Approach to Morphological Analyzer for Tamil Language. *International Journal on Computer Science and Engineering (IJCSE)*, Vol. 02, No. 06, 1944-195.

George, A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM* Vol. 38, No. 11:39-41.

Goldsmith, J. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27, 153–198.

Hernault, H., Bollegala, D., and Ishizuka, M. 2010a. A semi-supervised approach to improve classification of infrequent discourse relations using feature vector extension. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. *Association for Computational Linguistics*: Cambridge, MA, pp. 399-409.

Hernault, H., Predinger, H., Duverle, D.A., and Ishizuka, M. 2010. HILDA: A discourse parser using support vector machine classification. *Dialogue and Discourse*, vol.1, no.3, pp. 1-33.

Lin, Z., NG, H., and Kan, M. 2014. A PDTB-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2), 151-184.

Manfred Stede, Stergos Afantenos, Andreas Peldszus,Nicholas Asher, and Jeremy Perret. 2016. Parallel discourse annotations on a corpus of short texts. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Portoroz.

Mann, W.C. and Thompson, S.A., 1988. Rhetorical structure theory: Toward a functional theory of

text organization. *Text-interdisciplinary Journal for the Study of Discourse*, 8(3), pp.243-281.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from RST discourse parsing. In *EMNLP*, pages 2212–2218, 2015.

Polanyi, L.C., Culy, M.H., Van Den Berg, Thione, G.L., and Ahn, D. 2004. Sentential structure and discourse parsing. In *Proceedings of the ACL 2004 Workshop on Discourse Annotation*, Barcelona, Spain, pp. 80-87.

Sobha Lalitha Devi and B. N. Patnaik (2004), Discourse Connectives and Their Arguments in Malayalam", *24th South Asian Language Analysis*, University of Stony Brook , New York, November 19-21

Sobha Lalitha Devi, Lakshmi S and Sindhuja Gopalan(2014). "Discourse Tagging for Indian Languages", In *A. Gelbukh (ed), Computational Linguistics and Intelligent Text Processing*, Springer LNCS Vol 8403, pp. 469-480.

Sobha Lalitha Devi, Sindhuja Gopalan, Lakshmi S (2014). "Automatic Identification of Discourse Relations in Indian Languages", In *proceedings of 2nd Workshop on Indian Language Data: Resources and Evaluation*, Organized under LREC2014, Reykjavik, Iceland

Stamatatos, E. 2008. Author identification: Using text sampling to handle the class imbalance problem. *Information Processing and Management*, 44(2), 790–799.

Subalalitha, C.N., and Ranjani Parthasarathi 2013. A Unique Indexing Technique for Indexing Discourse Structures. *Journal of Intelligent Systems*. Volume 23, Issue 3, Pages 231–243.

Subalalitha, C.N., and Ranjani Parthasarathi. 2015. Building a Language-Independent Discourse Parser using Universal Networking Language. *Computational Intelligence*, 31: 593–618.

Subba, R., and Di Eugenio, B. 2009. An effective discourse parser that uses rich linguistic information. In *The Annual Conference of the North American Chapter of the ACL*, Boulder, CO, pp. 566-574.

Sucheta Ghosh, Giuseppe Riccardi and Richard Johansson 2012 Global features for shallow discourse parsing. In: *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 150–159

Sucheta Ghosh, Richard Johansson, Giuseppe Riccardi and Sara Tonelli 2011 Shallow discourse parsing with conditional random fields. In: *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011)*, pp. 1071–1079, Chiang Mai, Thailand, November

Sucheta Ghosh, Richard Johansson, Giuseppe Riccardi and Sara Tonelli 2012 Improving the recall of a discourse parser by constraint-based postprocessing. In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*

Uladzimir Sidarenka, Andreas Peldszus, and Manfred Stede. 2015. Discourse Segmentation of German Texts. *Journal of Language Technology and Computational Linguistics (JLCL)*, 30(1):71–98.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Yangfeng Ji, and Noah A. Smith. (2017). Neural discourse structure for text categorization. *arXiv preprint* arXiv:1702.01829.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse relation language models. In *NAACL-HLT*.

Introducing Aspects of Creativity in Automatic Poetry Generation

Brendan Bena
Drury University
900 N. Benton Ave.
Springfield, Missouri 65109

Jugal Kalita
UC-Colorado Springs
1420 Austin Bluffs Pkwy.
Colorado Springs, Colorado 80918

Abstract

Poetry Generation involves teaching systems to automatically generate text that resembles poetic work. A deep learning system can learn to generate poetry on its own by training on a corpus of poems and modeling the particular style of language. In this paper, we propose taking an approach that fine-tunes GPT-2, a pre-trained language model, to our downstream task of poetry generation. We extend prior work on poetry generation by introducing creative elements. Specifically, we generate poems that express emotion and elicit the same in readers, and poems that use the language of dreams—called dream poetry. We are able to produce poems that correctly elicit the emotions of *sadness* and *joy* 87.5 and 85 percent, respectively, of the time. We produce dreamlike poetry by training on a corpus of texts that describe dreams. Poems from this model are shown to capture elements of dream poetry with scores of no less than 3.2 on the Likert scale. We perform crowdsourced human-evaluation for all our poems. We also make use of the Coh-Metrix tool, outlining metrics we use to gauge the quality of text generated.

1 Introduction

Many natural language processing tasks require the generation of human-like language. Some tasks, such as image and video captioning and automatic weather and sports reporting, convert non-textual data to text. Some others, such as summarization and machine translation, convert one text to another. There are additional tasks that aim to produce text, given a topic or a few keywords such as story generation, joke generation, and poetry generation, among others.

Poetry generation produces creative content, and delivers the content in an aesthetically pleasing manner, usually following a specific structure. Thus, in addition to generating text as if in a story, the lines produced usually have a certain length, quite frequently there is a rhyming scheme as well as rhythm, and organization into structures such as couplets, quatrains, quintets, and stanzas. Among other tools, creativity comes from unusual usage of words through effects such as alliteration, assonance, and elision; use of metaphors, symbolism, and other linguistic devices; licensing of underlying imagery with expressed feelings, sentiments, and emotions.

Work in natural language generation can be traced to pioneering rule-based simulations of chatbots such as the "psychotherapist" Eliza (Weizenbaum et al., 1966) and paranoid schizophrenia-suffering PARRY (Colby, 1981). Surveys such as (Hovy, 1990; Reiter and Dale, 2000; Gatt and Krahmer, 2018; Santhanam and Shaikh, 2019) have described the progress in natural language generation over 50 years. Of late, the use of deep learning has produced enviable progress in natural language generation, especially in topics such as machine translation (Bahdanau et al., 2014; Wu et al., 2016), image captioning (Mao et al., 2014) and dialogue generation (Li et al., 2016).

This paper discusses the automatic generation of natural-sounding poems that are creative. Creativity comes in many hues, and we experiment with a few established ways of creative expression in poetry generation. First, we generate poetry that can potentially evoke a response from the readers or hearers in terms of emotions and feelings they generate. Additionally, we choose the idea of mimicking the language of dreams as another form of creative expression due to its longstanding history in poetry. Dream poetry dates back to medieval times where famous fourteenth century authors, like Chaucer, experimented using dreams as the structure for an image or picture they wished

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 26–35
Hyderabad, India, December 2019.

to paint with a poem (Spearing, 1976a). A dream poem is said to be characterized by the 'I' of the poem and its substance of a dream or a vision included (Lynch, 1998). To the best of our knowledge, prior work on poetry generation, whether using deep learning or not, has not explored the incorporation of emotion-eliciting phraseology or elements of creativity such as dream poetry.

Our research provides the following contributions:

- generating grammatical, coherent, and flowing poetry using the powerful and versatile GPT-2 architecture,
- successfully generating poetry that elicits certain emotions in readers, and
- generating poems that follow time-honored tradition of dream-like language usage and imagery.

This paper is organized as follows. Section 2 presents related work. Section 3 discusses our approach to creative text generation including pre-processing steps, architecture used, and approaches to training. Section 4 discusses our experiments and results. Finally, we present evaluation of our research in Section 5, followed by conclusions and future work in Section 6.

2 Related Work

Early methods for poetry generation made use of template-oriented and rule-based techniques. These approaches often required a large amount of feature picking and knowledge of syntactic and semantic rules in a language (Oliveira, 2009, 2012). Other methods treated poetry generation as special cases of machine translation or summarization tasks (Yan et al., 2013; He et al., 2012). We believe that forcing a model to adhere to specific rules or templates, or summarizing or translating a given text to generate new poetry is unlikely to lead to the artistically expressive quality we seek to generate.

More recently, deep learning methods have become prevalent in natural language generation, including poetry generation. Zhang and Lapata (2014) for instance, used Convolutional (CNN) and Recurrent Neural Networks (RNN) to generate Chinese Poetry. RNNs allow for short-term memory of the language to be maintained by inputting the generated output of a network cell back into itself, essentially building context. Ghazvininejad et al. (2017) used Long Short-Term Memory (LSTM) units, which are advanced gated versions of RNNs, to the task of poetry generation. Wei et al. (2018) attempted to address the style issue by training the networks using particular poets and controlling for style in Chinese poetry. They found that with enough training data, adequate results could be achieved. Problems related to poetic structure were addressed by Hopkins and Kiela (2017). They generated rhythmic poetry by training the network on only a single type of poetry to ensure produced poems adhered to a single rhythmic structure. It was found in human evaluations that while the poems produced were rated to be of lower quality than human produced poems, they were indistinguishable from human produced poems. Lau et al. (2018) took the LSTM approach one step further with the *Deepspeare* model by employing an attention mechanism to model interactions among generated words. They also use three neural networks, one for rhythm, one for rhyming and another for word choice in their quest to generate Shakespeare-like sonnets.

Vaswani et al. (2017) developed a deep neural architecture called the Transformer that did away with any sort of need for recurrence. The Transformer also employed an elaborate attention mechanism that has been shown to be useful in natural language tasks. Radford et al. (2019) used this architecture in their Generative Pretrained Transformer 2 (GPT-2) model. GPT-2 is capable of many downstream tasks like text generation but to our knowledge, research has not been published using the GPT-2 model specifically for poetry generation.

On a slightly different but related note, natural language generation influenced by multi-modal input was attempted by Vechtomova et al. (2018) to generate song lyrics in the style of specific artists by fusing outputs coming from lyrical inputs processed by an RNN and audio clips processed by a CNN. Text generation has also been influenced, in a cross domain manner, through images. The works of Liu et al. (2018) have shown that coupled visual-poetic embeddings can be used to pick out poetic clues in images, which in turn can be used to inspire the generated text. Though influenced natural language generation in and of itself is not a novel idea, we feel our attempt to style text with the intent of eliciting particular emotions provides a creative way to explore this subtask.

3 Approach

Our goal is to successfully demonstrate the introduction of creative flair in automatic poetry generation in two exemplar ways: explicit show of emotion and the use of language that is predominantly first person with dream-like imagery. To enable the expression of emotion in generated poems, our work involves a preliminary step of scoring a corpus of downloaded poems for emotion to produce subsets of poems that express one of eight different identified emotions. This step is followed by the actual generation of poems by fine-tuning the pre-trained GPT-2 natural language model. We train eight separate models for eight different emotions, each on a sub-corpus predominantly demonstrating a particular emotion. To generate poems that use dream-like language, we create a text corpus composed of a large number of dream transcriptions created in first person by actual viewers of dreams. In this case, we apply transfer learning by fine-tuning the pre-trained GPT-2 on the dream corpus, followed by training again on poetry. We evaluate the generated poems using automated techniques as well as humans.

3.1 Poem Emotion Scoring

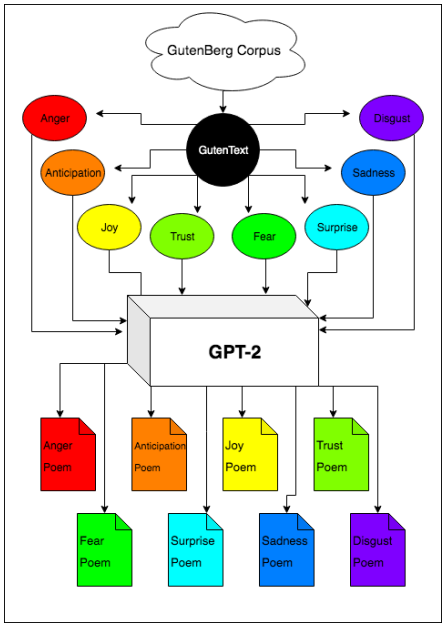


Figure 1: A high-level overview of our project implementation for emotion eliciting poetry

A high-level overview of the emotion elicitation portion of our project is shown in Figure 1. To create a corpus of poems based on the emotions they elicit, we make use of the EmoLex dictionary (Mohammad and Turney, 2013). EmoLex is a word-level emotion lexicon that associates English words with the eight different emotion categories we wish to explore. Each poem (or book of poems) in our dataset is given a score that is the total of the associated emotion scores in EmoLex for each word. The maximum emotion word score is taken and the poem is labeled under that emotion category. We create eight such datasets, one corresponding to each emotion category supported by EmoLex. This approach allows us to to train multiple models on our split dataset.

Currently, the emotions of *joy, anticipation, trust, anger*, and *sadness* represent a large portion of our data while the emotions of *surprise, disgust,* and *fear* are severely underrepresented. Table 1 shows key differences in models including the number of tokens in the text and the final average loss during training.

3.2 GPT Architecture

To create a model for poetic language, we propose finetuning OpenAI's GPT-2 architecture. GPT-2 is a Transformer-based model that was trained simply to predict the next word in a 40GB text corpus (Radford et al., 2019). This 40GB dataset, *WebText*, was scraped from the internet with certain heuristics that aimed to gather only quality text (i.e. only outbound Reddit links from posts with a karma rating of 3 stars or better). By training on such a large, all-encompassing corpus of text, the architecture has proven to model the English language well and has obtained state-of-the-art results on downstream text-based tasks such as machine translation, question answering, and summarization. We leverage GPT-2's pre-trained knowledge of language for our downstream task of peotry generation.

GPT-2 is the successor of OpenAI's first Transformer-based architecture, GPT (Radford et al., 2018), with a few changes to the structure. The medium version of GPT-2 we use contains 345M parameters and is a 24 layer, decoder-only Transformer architecture. GPT-2 moves layer normalization to the input of each sub-block, adds another layer normalization after the final self-attention block and increases context size from 512 to 1024 tokens. This architecture allows for long term dependencies to be captured better in language modeling. GPT-2's attention mechanism is referred to as a masked multi self-attention head.

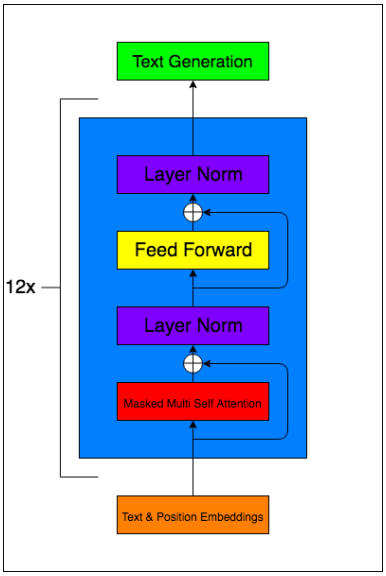


Figure 2: GPT Architecture. Adapted from (Radford et al., 2018, 2019)

Data	Model Size	# of Tokens	Final Loss
anger	345M	1,292,457	0.27
antici-pation	345M	2,314,637	1.30
joy	345M	11,668,792	3.19
sadness	345M	2,090,915	1.03
trust	345M	16,667,178	3.39

Table 1: Comparison of 5 emotion models trained.

This technique allows for a relationship to be modeled for all words in an input sequence. Words that have multiple meanings can then be represented based on the context they appear in. Higher attention scores from surrounding words relate to a larger contribution to the representation of a word. GPT-2 makes use of byte-pair encoding (BPE) like its predecessor GPT but on UTF-8 byte sequences (Sennrich et al., 2015). GPT-2's encoding is somewhere in between character level and word level. The model also prevents different versions of common words from being duplicated (i.e. *fate!*, *fate?*, and *fate* would not be joined). This technique improves the quality of the final byte segmentation. GPT-2's encoding rids the need for pre-processing or tokenization of data and is able to assign a probability to any Unicode string.

3.3 Training for Creative Poem Generation

The task-agnostic nature of GPT-2 allows us to take a fine-tuning approach to our downstream task of poetry generation. Our approach to generating poems that exhibit emotion as well as dream-like imagery involves training the pre-trained GPT-2 model. Our training protocol for the two cases are stated briefly below.

3.3.1 Generating Emotion Poems

Poetry is a personal form of writing that expresses human feelings, and Mill (1860) famously said "What is poetry, but thoughts and words in which emotion spontaneously embodies itself?" Mill (1833) also said "The object of poetry is confessedly to act upon the emotions". Expressing emotions, with possible motive of eliciting the same emotions in readers, is a basic characteristic of poems. Our goal in this paper is to use artificial neural networks to generate poems that explicitly evoke certain specific emotions.

To generate poems with emotional content, we have split our poetry data into sub-corpora, one sub-corpus for each emotion. We train the already pre-trained GPT-2 on a sub-corpus of poems that demonstrate a certain emotion. Pre-trained GPT-2 has a very strong foundational knowledge of English. We find that training it again on emotion-bearing poetry seems to enable it to generate high quality poetry, which is even able to use emotion-laden words for the correct form of elicitation. We also find that the poems we generate seem to exhibit proper punctuation as well as lines that have poem-appropriate length and sentences that are grammatically correct. In addition, the poems we generate seem to be quite readable and demonstrate high coherence. Detailed analyses are reported in the next section.

3.3.2 Generating Dream Poems

Dream poems represent a style of poetry that was "astonishingly" popular in the 14th through the 16th centuries (Spearing, 1976b; Windeatt, 2003) and are still popular (Russo, 2003). Such poems tell a story based on a dream or a number of dreams, dreamt by the narrator or by a character that the poet introduces. Spearing (1976b) claimed that dream poems are based on objective experience, but at the same time they are free of constraints of everyday possibilities. Such poems represent the outcome of a poetic process with many different influences, models, and analogues (Windeatt, 2003), but without going into such details, our goal is to see if an ANN can produce poems which share characteristics with dream po-

ems.

To generate poems that demonstrate first-person language with dream-like imagery, we take a similar approach. However, in this case, GPT-2 undergoes three separate training cycles. The first cycle is the pre-training that GPT-2 goes through before release to the public by OpenAI. Second, we train the pre-trained model on a corpus of first-person dream descriptions. Third, we train again on poems. Our hypothesis is that pre-training by OpenAI results in good basic knowledge of English; that training on the dream corpus endows the network with the knowledge of first-person imagery-based language; and that the last training cycle teaches the network language of poems. We demonstrate in the next section that we are not far off from our being successful in our hypothesis.

3.4 Text Generation and Sampling

As stated by Radford et al. (2019), the core approach of GPT-2 is language modeling. A language model can be thought of as a probability distribution over a sequence of words in the form:

$$p(w_1, ..., w_n). \quad (1)$$

Likewise, natural language tends to have a sequential order so it can be modeled in terms of the conditional probability of a word given the words preceding it (Bengio et al., 2003):

$$p(w_n|w_1, ..., w_{n-1}). \quad (2)$$

We make use of this probabilistic style of language modeling by sampling from the distribution in a semi-random fashion. Just as the GPT-2 paper does for its text generation, we make use of Top K sampling, limiting the possible guesses of words to 40. In addition to Top K, we make use of a temperature constant of 0.75 which controls randomness in the distribution. A temperature closer to 0 correlates to less randomness while a temperature closer to 1 relates to more randomness. Finally, at the end of the generation process, we employ a simple text cleaning algorithm that allows poems to end more naturally and rather than trail off as they do sometimes.

4 Experiments and Results

4.1 Datasets and Resources

In order to classify emotion-eliciting poems or books, we use the NRC Word-Emotion Association Lexicon (EmoLex) resource. EmoLex[1] was created by the National Research Council of Canada and includes 14,182 English words that are associated with different emotions and positive or negative sentiment (Mohammad and Turney, 2013). Words in EmoLex have been manually annotated via crowd-sourcing and emotions fall into one or more categories of eight basic emotions: *joy, trust, fear, surprise, sadness, anticipation, anger, and disgust* (Plutchik, 2014). We elect to use this simplified version of the Wheel of Emotions due to its parallels with the available EmoLex dataset. This resource provides us with a way to fabricate a ground truth in the types of emotion-infused texts we wish to use for training data.

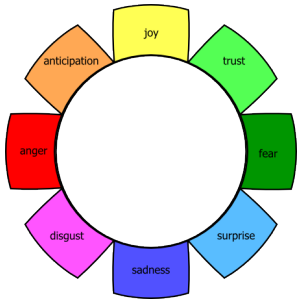


Figure 3: American pyschologist Robert Plutchik's Wheel of Emotions

To handle the training and generation portions of the project, we draw data from the Project Gutenberg website[2]. Project Gutenberg is a massive online database containing over 59,000 eBooks. We limit this corpus to a smaller subcorpus using an adaptation of the GutenTag tool (Brooke et al., 2015). This tool allows us to place constraints on the amount of literature we choose to use in our work. Our final dataset includes approximately three million lines of poetic text from the Gutenberg database and is further divided by poem/book into our eight emotion categories.

We attempt to create dream poetry by making use of the *DreamBank* dataset. The *DreamBank* was created by Schneider & Domhoff at UC-Santa Cruz[3]. The dataset contains a collection of over 20,000 dreams from users age 7 to 74. We scraped this dataset from the website assuring that dreams collected were recorded only in English. The *DreamBank* allows us to attempt trans-

[1] https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm
[2] https://www.gutenberg.org/
[3] https://www.dreambank.net/

Heard I a song of joy,
A song of happy sound,
Fills all the air I breathe,
To him I sing, to him
I sing the happy song.
All night long on the steep green grass
I ride and sing

Figure 4: A hand-picked, automatically generated poem from the joy model

The other, who with one accord
Wrote my essay, in that he was dear
And good, and knew well, how we ought to treat
A man of such renown, and such love?
He's a good honest man, no doubt

Figure 5: A hand-picked, automatically generated poem from the trust model

fer learning by finetuning on the dream dataset first, then further finetuning on our poetry dataset.

Initially, we retrained 6 GPT-2 based models. Default training parameters were used each of the 5 different emotion datasets and our dream dataset. All were trained for 12,000 steps (except for our dream model which was trained for 12k steps on dreams and on poetry) with a learning rate of 0.0001. When generating text, we do not input context: we allow the model to write the poem entirely through the sampling of conditional probability from the language it has modeled.

Figures 4 through 8 give examples of 5 poems that we have hand-picked to illustrate the quality of poems generated. A cursory glance at the poems reveals the high quality of the text in terms of lexical choice, grammatical integrity, and semantic cohesion. We discuss how we quantitatively assess the poems below.

5 Evaluation

In the first crowd-sourced analysis of our emotion-eliciting poetry we presented four poems from each category (of the five data-represented emotion categories) to ten human reviewers with undergraduate level educational backgrounds. All reviewers are native speakers of English. Poems presented were randomly selected from the top 20 EmoLex scored poems out of a pool of 1,000 generated poems. These reviewers were asked to rate each poem based on the emotions elicited within

We have reached the peak of the highest mountain
in the world
The mountain of dreams.
This is the view
Across the valley,
One hour's journey back,
We crossed it on the way between
A band of beautiful young women.
There was

Figure 6: A hand-picked, automatically generated poem from the anticipation model

A long trail of falling mist
Had made its way here, and now
Aerily it seemed, as if to drown
The discordant thunder clang.
It seemed to drown the music of the rain;
In this lost place of sorrow
Far off

Figure 7: A hand-picked, automatically generated poem from the sadness model

Amidst the chaos throng'd, with angry voices each
His rival's mockery; loud their scorn was fill'd;
So fierce their rage, and in their eager power
Met on the walls of Troy, were fill'd with dismay.

Figure 8: A hand-picked, automatically generated poem from the anger model

A thousand stars at once,
An hundred thousand stars!
The sun was low,
And the stars were bright,
My heart would do the same.
A thousand stars at once,
A hundred thousand stars!
The night had begun,
And the stars were all the same.
When I came back from the dead,
I saw the stars

Figure 9: A hand-picked, automatically generated poem from the dream model

For she was mine.
I was the only one
She had,
And a thousand other friends,
And a hundred more
She held me dear.
Her eyes were clear, her cheeks were bright,
Her heart was like a rose,
Her mouth was full of music,
Her lips were white
As snow,
And the music she sang

Figure 10: A hand-picked poem, automatically generated from the dream model

Emotion	Anger	Antic.	Joy	Sad.	Trust
%	65	40	85	87.5	32.5

Table 2: Average percentage of correctly elicited emotion across four poems in each category

them after reading. An emotion was deemed correctly elicited if the associated Likert score was 4 or greater from the reviewer. Table 2 illustrates the results from our evaluation. When taking the average percentage of correct emotion-eliciting poems, the models of joy, sadness, and anger produced the most promising results while the trust and anticipation models were less than satisfactory. We believe this is because joy, sadness and anger are basic or fundamental emotions compared to trust and anticipation, which are more complex and difficult to explain. Although there are many opinions among psychologists about what constitute basic emotions, joy, sadness and anger, (especially the last two) seem to occur the most often in proposals that demarcate a set of basic emotions (Ortony and Turner, 1990).

To preserve consistency in our experiments, we evaluate our dream model poetry in a manner similar to our evaluation of the emotion poems. Four poems from the model were presented to the same ten judges and they were asked to assess the poems based on qualities of dream poetry. These poems were cherry picked from a pool of 1,000 generated poems. A dream poem is said to have the following qualities (Windeatt, 2003; Spearing, 1976b; Russo, 2003) among many other qualities. We believe these three are the least ambiguous and easiest to decipher for human evaluation.

Poem	1	2	3	4
Qual 1	**5**	4.9	4.8	4.5
Qual 2	3.5	**4.1**	3.2	3.3
Qual 3	3.9	**4.2**	3.7	3.7

Table 3: Average Likert score of users for each poem

- Quality 1: The poem is generally a first-person expression
- Quality 2: The poem's main substance is dream or vision like
- Quality 3: The poem recounts or foretells an experience or event

Analysis of results show that machine generated poems are able to capture the first person perspective well, achieving between 4.5 and 5 average Likert scores. The poems often appear to retell a story or an event, scoring between 3.7 and 4.2 average Likert scores. The nature of poetry and dream recounts that make up our data is often narrative, so this result stands to reason. However, Quality 2 scores of the poem substance containing a dream or vision are questionable. We suspect the Quality 2 score is lower due to the ambiguity in ascertaining dream text from regular text. Table 3 highlights our results for the dream model.

Currently, there exists no widely available standard for evaluating poetry generation. Scores like BLEU, ROUGE, METEOR, etc. are more suited for Machine Translation (MT) tasks (Zhang et al., 2019). For example, they compare how similar sentence P is to translated-sentence $\hat{P}$. Instead, we outline some metrics from the Coh-Metrix web tool that helps us further quantitatively evaluate the quality of text generated. With the goal of eliciting emotions, we claim that subjective analysis of generated poetry is superior to any available objective metrics.

5.1 Coh-Metrix

To provide a quantitative calculation of the caliber of text our models produce, we outline in this section relevant metrics from the University of Memphis Coh-Metrix tool (Graesser et al., 2004). Coh-Metrix is a text evaluation software kit and from it, we have chosen 8 forms of assessment. The first two, Flesch-Kincaid Grade Level (FKGL) and Flesch Reading Ease (FRE), are two standard measures that deal with text readability and ease (Klare, 1974). The FKGL scores a text from grade level 0 to 18, while the FRE score is a 0-100 index

Model	FRE	FKGL	IMGc	CNCc	LDTTRa	PCREFp	PCSYNp	PCNARp
anger	93.07	2.01	445.91	407.16	0.53	0.68	80.78	53.19
anticipation	100	0.83	440.93	403.10	0.40	7.78	83.65	81.86
joy	100	0.39	446.23	403.07	0.39	11.90	91.31	78.52
sadness	98.20	1.18	444.96	403.25	0.44	1.88	88.69	72.91
trust	100	0.16	434.66	412.72	0.33	18.14	84.61	91.31
dream	100	0	427.36	377.48	0.24	99.90	65.17	70.88

Table 4: Average Coh-Metrix evaluations across 25 randomly selected poems from each model.

with 100 being an easily readable text. We aim to produce text that is readable by all, so a low FKGL score and high FRE score would be ideal.

The next metrics we employ evaluate at the word level. The word imageability (IMGc) and word concreteness (CNCc) scores measure content words on their ability to create an image in the reader's mind and their ability to appeal to a reader's senses, respectively (Coltheart, 1981). We aim for our art to create a connection between the reader and poem, so we believe imageability and concreteness of content words are two good measures with this in mind. We also make use of three text easibility principal component scores: narrativity (PCNARp), referential cohesion (PCREFp), and syntactic simplicity (PCSYNp) (Graesser et al., 2004). The text easibility PC scores are percentile scales, and thus we aim for higher numbers for these scores. Finally, we make use of the Lexical Diversity Type:Token Ratio score (LDTTRa) for all words. LDTTRa measures the ratio of *type* (unique) words to all *tokens* in the text. Because our text is relatively short, we aim for a middle ground in the LDTTRa ratio, meaning there is uniqueness in the word choice of the text, but cohesion is still upheld.

Inspection of our Coh-Metrix results show that randomly selected poems from all models fall at or below the 2nd-grade reading level (in FKGL scores) and are greater than 93 on the FRE scale. This suggests generated poems are easily readable by the majority of viewers. Looking at the IMGc and CNCc scores, we see that our poems, except for the dream model concreteness, fall in the 400s. Words with higher imageability and concreteness fall around the low 600s while words that are lower fall around the upper 200s on this scale. These scores reveal that our models are generating text that is concrete in word choice and that paint a picture. Our dream model scoring lower in the concreteness is reasonable as the word choice of dreams tends to be more abstract. Lastly, percentile scores of PCSYNp and PCNARp show that the majority of models are producing poems that are both syntactically simplistic and narrative. Most PCREFp scores are on the lower end of the scale. We suspect the reason these scores are lower is because the poems are not necessarily related and were all input at once. Table 4 highlights these scores for each poetry model.

6 Conclusion & Future Work

In this paper we influenced automatic natural language generation to create poetry through the use of classified emotion poems and dream text. To do so, we first leveraged a word-level emotion lexicon to construct a meaning for emotion-eliciting text and used that text to train separate language models. Next, we gathered data of dream records and employed transfer learning in attempts to generate dream-like poetry. The work reported in this paper seeks to create art in the form of auto-generated poetry while opening the door to more projects involving emotion-eliciting text-based tasks and influenced creative neural generation.

We would like to thank the reviewers for their feedback on this project. Comments and suggestions from reviewers — both those that the were incorporated into this article and those on which we will report in future work — provide invaluable insight as to improving our results. Importantly, our continuing research involves gathering a more comprehensive human-evaluation with a larger number of reviewers and poems to be judged. We also wish to gather data for the underrepresented emotion categories, leading, ideally, to a more robust language model for each emotion. Our work thus far provides a baseline for introducing emotions into generated text via a word-level lexicon, but we wish to employ other

tools — segment-level lexicons, for example — in an attempt to better capture the contextual dependencies of emotion. Additionally, the word-level baseline we have produced focuses on generating single-emotion text. We are interested in examining poems of multiple emotions and different levels of intensity to expand on this study. Finally, we wish to seek out additional forms of replicating creativity that artists incorporate in their work.

References

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

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

Julian Brooke, Adam Hammond, and Graeme Hirst. 2015. Gutentag: an nlp-driven tool for digital humanities research in the project gutenberg corpus. In *Proceedings of the Fourth Workshop on Computational Linguistics for Literature*, pages 42–47.

Kenneth Mark Colby. 1981. Modeling a paranoid mind. *Behavioral and Brain Sciences*, 4(4):515–534.

Max Coltheart. 1981. The mrc psycholinguistic database. *The Quarterly Journal of Experimental Psychology Section A*, 33(4):497–505.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada. Association for Computational Linguistics.

Arthur C. Graesser, Danielle S. McNamara, Max M. Louwerse, and Zhiqiang Cai. 2004. Coh-metrix: Analysis of text on cohesion and language. *Behavior Research Methods, Instruments, & Computers*, 36:193–202.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating chinese classical poems with statistical machine translation models. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Jack Hopkins and Douwe Kiela. 2017. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 168–178.

Eduard H Hovy. 1990. Pragmatics and natural language generation. *Artificial Intelligence*, 43(2):153–197.

George R Klare. 1974. Assessing readability. *Reading research quarterly*, pages 62–102.

Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. 2018. Deep-speare: A joint neural model of poetic language, meter and rhyme. *CoRR*, abs/1807.03491.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Bei Liu, Jianlong Fu, Makoto P. Kato, and Masatoshi Yoshikawa. 2018. Beyond narrative description: Generating poetry from images by multi-adversarial training. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, pages 783–791, New York, NY, USA. ACM.

Kathryn L. Lynch. 1998. *Medieval Dream-Poetry*. Cambridge University Press.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2014. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*.

John Stuart Mill. 1833. What is poetry? *Monthly Repository*, 7(73):60–70.

John Stuart Mill. 1860. Thoughts on poetry and its varieties. *The Crayon*, 7(5):123–128.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. 29(3):436–465.

Hugo Oliveira. 2009. Automatic generation of poetry: an overview. *Universidade de Coimbra*.

Hugo Gonçalo Oliveira. 2012. Poetryme: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21.

Andrew Ortony and Terence J Turner. 1990. What's basic about basic emotions? *Psychological review*, 97(3):315.

Robert Plutchik. 2014. *Emotions*. Psychology Press.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Richard A Russo. 2003. Dream poetry as dream work. *Dreaming*, 13(1):13–27.

Sashank Santhanam and Samira Shaikh. 2019. A survey of natural language generation techniques with a focus on dialogue systems-past, present and future directions. *arXiv preprint arXiv:1906.00500*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

A. C. Spearing. 1976a. *The High Medieval Dream*. Stanford University Press.

Anthony Colin Spearing. 1976b. *Medieval dream-poetry*. CUP Archive.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Olga Vechtomova, Hareesh Bahuleyan, Amirpasha Ghabussi, and Vineet John. 2018. Generating lyrics with variational autoencoder and multi-modal artist embeddings. *CoRR*, abs/1812.08318.

Jia Wei, Qiang Zhou, and Yici Cai. 2018. Poet-based poetry generation: Controlling personal style with recurrent neural networks. In *2018 International Conference on Computing, Networking and Communications (ICNC)*, pages 156–160. IEEE.

Joseph Weizenbaum et al. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Barry Windeatt. 2003. Literary structures in chaucer. *The Cambridge Companion to Chaucer*, pages 214–232.

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*.

Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. 2013. I, poet: automatic chinese poetry composition through a generative summarization framework under constrained optimization. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with BERT. *CoRR*, abs/1904.09675.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.

A Language Invariant Neural Method for TimeML Event Detection

Suhan Prabhu, Pranav Goel, Alok Debnath and **Manish Shrivastava**
International Institute of Information Technology
Hyderabad, Telangana, India
{suhan.prabhuk, pranav.goel, alok.debnath }@research.iiit.ac.in
m.shrivastava@iiit.ac.in

Abstract

Detection of TimeML events in text have traditionally been done on corpora such as TimeBanks. However, deep learning methods have not been applied to these corpora, because these datasets seldom contain more than 10,000 event mentions. Traditional architectures revolve around highly feature engineered, language specific statistical models.

In this paper, we present a Language Invariant Neural Event Detection (ALINED) architecture. ALINED uses an aggregation of both sub-word level features as well as lexical and structural information. This is achieved by combining convolution over character embeddings, with recurrent layers over contextual word embeddings. We find that our model extracts relevant features for event span identification without relying on language specific features.

We compare the performance of our language invariant model to the current state-of-the-art in English, Spanish, Italian and French. We outperform the F1-score of the state of the art in English by 1.65 points. We achieve F1-scores of 84.96, 80.87 and 74.81 on Spanish, Italian and French respectively which is comparable to the current states of the art for these languages. We also introduce the automatic annotation of events in Hindi, a low resource language, with an F1-Score of 77.13.

1 Introduction

Automatic extraction of events has gained sizable attention in subfields of NLP and information retrieval such as automatic summarization, question answering and knowledge graph embeddings (Chieu and Lee, 2004; Glavaš and Šnajder, 2014), as events are a representation of temporal information and sequences in text. Various developments in guidelines and datasets for event detection have been met with equally fast paced evolution of automatic event annotation and detection methodologies in the last few years (Doddington et al., 2004; Pustejovsky et al., 2010; O'Gorman et al., 2016). On a larger scale, event extraction has extended to many languages beyond English, including French (Bittar et al., 2011), Spanish (Saurı, 2010), Italian (Caselli et al., 2011a) and very recently, Hindi (Goud et al., 2019b). Event detection architectures have their origins in statistical models such as K-means and hierarchical clustering methods (Arnulphy et al., 2015), which have more recently given way to neural models. Deep neural architectures on event annotation vary based on the approach taken to identifying and handling the data.

However, event detection as a problem shifts when we move away from the annotation paradigm of datasets such as ACE (Doddington et al., 2004) and TAC KBP (Mitamura et al., 2015) to TimeML datasets such as TimeBank (Pustejovsky et al., 2006), which are used in this paper. There has been limited use of deep learning methods on TimeBanks due to fewer event mentions and a need for data augmentation and bootstrapping. However, in this paper, we show that using subword level information, a language invariant deep learning model can provide similar event detection accuracies as heavily feature engineered language specific statistical methods without using any augmented data.

This paper has two main contributions. First, we introduce our model, the Architecture for Language Invariant Neural Event Detection (ALINED), which is a deep learning model for event extraction from TimeML event annotated datasets from five languages. We show that for four of these languages, using no augmented data, we achieve comparable F1 score on these datasets to heavily feature engineered language specific

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 36–44
Hyderabad, India, December 2019.

statistical models, with less than 12,000 event mentions in each. Secondly, to the best of our knowledge, we present the first ever baseline for neural event detection in Hindi using this model. Our architecture uses both word and character embeddings and captures information from them distinctly, before combining them into a coherent representation of both. This is then used to determine the label for each input word. The proposed architecture is language invariant as well, such that no part of the system undergoes a change when training on different languages. In presenting this architecture, we highlight the importance of using subword level information in order to incorporate morphological as well as syntactic features in event extraction. This can also be extended to other semantically oriented sequence labeling tasks

2 Related Work

Neural approaches to sequence tagging are common due to extensive developments in named entity recognition. Huang et al. (2015) introduced and cultivated the use of bidirectional LSTMs to incorporate features that could be used for sequence tagging using a CRF. Ma and Hovy (2016)'s architecture and the NeuroNER program (Dernoncourt et al., 2017) provided a basic architecture and influenced multiple developments to most sequence labeling tasks, including event detection and extraction (Araki, 2018). The task of event extraction in any language involves the identification of the event nugget (Ahn, 2006). Prominent work has been done to analyze the lexical and semantic features of event representation (Li et al., 2013), which served as a basis for neural event nugget detection (Liang et al., 2017).

The task of neural event detection has been attempted using a combination of networks, but mostly revolving around the use of convolutional neural architectures. Work in this approach focused on various aspects such as max-pooling to retrieve the structure of event nugget information (Nguyen and Grishman, 2015), modeling the skip-gram architecture to learn lexical feature representations (Chen et al., 2015) as well as using dynamic CNNs in order to extract lexical and syntactic features in parallel (Nguyen and Grishman, 2016). Recurrent neural architectures have also been employed for this task, which predict the location of the trigger based on combining the forward and backward features of sentences in which events occur (Nguyen et al., 2016; Ghaeini et al., 2016). Note that in both cases architectures focused on dealing with structural, lexical and contextual features.

In the domain of multi-lingual and cross lingual event detection, Feng et al. (2018) uses a combination of both LSTMs and CNNs for creating a language independent architecture for capturing events, while Goud et al. (2019a) used stacked RNNs for sequence labeling and a language discriminator to learn language features. The latter architecture implements the use of the character embeddings, but does not identify the relevant features independent of the word embeddings.

3 Model Description

In this section, we describe the ALINED model for the event detection. Primarily, we focus on how to capture event representation at both a character and a word level. In this model, we had to focus on the following major considerations:

1. Syntactic and lexical information captured by previous event detection tasks should be accounted for.

2. Furthermore, sub-word information is essential as morphological features are also useful in identifying event semantics if the language is morphologically rich, or has a free word order structure.

Fundamentally, our architecture generates character embeddings through convolution and aggregates this information using bidirectional LSTMs (Hochreiter and Schmidhuber, 1997). The same is done over pretrained word embeddings in parallel, creating distinct intermediate representations. These representations are combined using a highway architecture for a final representation, which is used for the sequence tagging task.

3.1 Generating Contextual Character Embedding

In order to generate character embeddings from the input sentence, we first use a CharCNN (Kim et al., 2016). Let $\mathcal{C}$ be the dictionary of all the characters in the language and $\mathcal{V}$ be all the words in the language. We first define the character embeddings matrix $\boldsymbol{E} \in \mathbb{R}^{d \times |\mathcal{C}|}$, where d is the dimensionality of the character embeddings, with the constraint that $d < |\mathcal{C}|$. Let word $w_i \in \mathcal{V}$

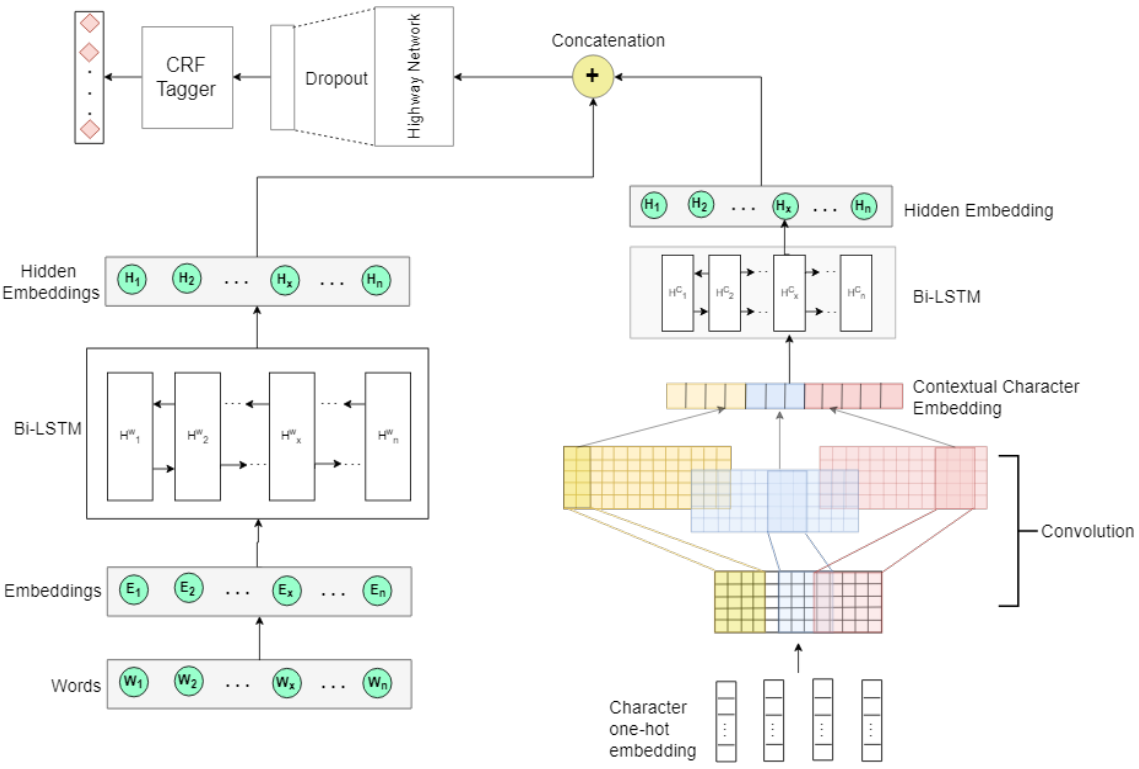


Figure 1: The proposed ALINED model

be made up of n characters, such that $\boldsymbol{c}^{w_i} = [c_1^{w_i}, c_2^{w_i}, \ldots, c_n^{w_i}]$. The character representation of w_i is therefore given by $\boldsymbol{E}^{w_i} \in \mathbb{R}^{d \times n}$.

We define a filter $\boldsymbol{W} \in \mathbb{R}^{d \times b}$ where b is the width of the filter. We apply a narrow convolution between $\boldsymbol{E}^{w_i}$ and $\boldsymbol{W}$, to obtain the embedding of w_i as:

$$e_i^{w_i} = f(\boldsymbol{W} \cdot \boldsymbol{E}^{w_i}[*, i : i + b - 1]) + b \quad (1)$$

where $\boldsymbol{E}^{w_i}[i : i + b - 1]$ accounts for all the characters of given window size of the word. The obtained embedding $e^{w_i} \in \mathbb{R}^{n-b+1}$. The function f is a non-linear function such as a hyperbolic tangent or a sigmoid. It is applied over the Frobenius inner product of the filter and the embedding value as $\boldsymbol{A} \cdot \boldsymbol{B} = \text{Tr}(\boldsymbol{A}\boldsymbol{B}^T)$ for any two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

We use max-pooling over the output embedding (instead of mean-pooling as it better incorporates the nature of natural language sequences (Xiang et al., 2016)) as:

$$w_i^c = \max_i e_i^{w_i} \quad (2)$$

For a total of h filters, each of varying widths, we get different representations of w_i. Therefore $\boldsymbol{w}_i^c = [\boldsymbol{w}_1^c, \boldsymbol{w}_2^c, \ldots, \boldsymbol{w}_h^c]$ is the representation of the ith word.

The aggregated word representations based on character information now capture the features that represent the event semantics at a sub-word level accurately. However, the contextual information has not been accounted for yet. This is done by using a bidirectional LSTM, as mentioned above.

$$h_i^c = \text{bi-LSTM}(\boldsymbol{w}_i^c, h_{i-1}^c, h_{i+1}^c) \in \mathbb{R}^{k \times l} \quad (3)$$

The bi-LSTM hidden state vector $\boldsymbol{h}^c = [h_1^c, h_2^c, ..., h_k^c]$, each h_i^c of dimension $\mathbb{R}^l$ is now propagated to the rest of the network. $\boldsymbol{h}^c$ can be seen as a lexically context-aware character representation of the words of the input sentence.

3.2 Using Contextual Word Embeddings

To capture structural information well, we use contextual word embeddings. Let $\boldsymbol{w} = [w_1, w_2, ..., w_k]$ be the words in a sentence. Let their corresponding pre-trained word embeddings be $\boldsymbol{e}^w = [e_1^w, e_2^w, ..., e_k^w]$. We aggregate the meaning of the sentence by passing the word embeddings through a bidirectional LSTM layer, as follows:

$$h_i^w = \text{bi-LSTM}(\boldsymbol{e}_i^w, h_{i-1}^w, h_{i+1}^w) \in \mathbb{R}^{k \times l} \quad (4)$$

Now each hidden state of $\boldsymbol{h}^w = [h_1^w, h_2^w, ..., h_k^w]$, i.e., each h_i^w of dimension

$\mathbb{R}^l$, is used in the rest of the network. Since the pre-trained word embeddings are already contextual in nature, we do not process it further. Note that $\boldsymbol{h}^w$ can be seen as the semantically context-aware representation of the words of the input sentence. This also includes the structure of event representation in that sentence.

3.3 Combining Character and Word Representations

Given the representations of the hidden states from characters and words, we combine the two using a concatenation function followed by a highway network. The concatenation is represented as follows:

$$h_i = f(h_i^w, h_i^c) \tag{5}$$

The function $f(\cdot)$ is the concatenation function, which can be represented as:

$$f(h_i^w, h_i^c) = \begin{cases} h_i^w \odot h_i^c & (6) \\ \boldsymbol{W} \cdot h_i^w \odot (1 - \boldsymbol{W}) \cdot h_i^c & (7) \\ \boldsymbol{W}^w \cdot h_i^w \odot \boldsymbol{W}^c \cdot h_i^c & (8) \end{cases}$$

Equation 6 is a direct concatenation of the hidden states $\boldsymbol{h}^c$ and $\boldsymbol{h}^w$. A direct concatenation automatically implies that the information gathered from the representations are given equal weight. However, this is not true for all languages, as languages with fewer inflections require less information from the character representations and more from the word representations.

Equations 7 and 8 attempt to account for this by using a shared weight concatenation and a weighted concatenation respectively. In equation 7, $\boldsymbol{W} \in \mathbb{R}^{k \times k}$ is a weight matrix, where the values are scaled down to 1, in order to capture the relative importance of each h_i^c and $h_i^w \; \forall h_i^c \in \boldsymbol{h}^c, h_i^w \in \boldsymbol{h}^w$. This shared weighting is a modification of the concept of *leaky integration* (Bengio et al., 2013). On the other hand, equation 8 uses two independent weight matrices, $\boldsymbol{W}^c, \boldsymbol{W}^w \in \mathbb{R}^{k \times k}$, which does not constrain the network to use on other the other hidden representation. However, the gradients are still clipped at a low value (≈ 1) to avoid explosion.

We then use the highway network (Srivastava et al., 2015) on the combined hidden state vector $\boldsymbol{h}$. This network adaptively "carries" some dimensions of $\boldsymbol{h}$ to the output for predicting the correct label sequence. Therefore, the hidden states undergo the following transformation (Wen et al., 2016):

$$h_i = \rho(h_i) \odot g(\boldsymbol{W}_H \cdot \bar{h}_i + b_H) + (1 - \rho(h)) \odot \bar{h}_i \tag{9}$$

The function $\rho(h^w) = \sigma(\boldsymbol{W}_\rho \cdot h_i + b_\rho)$, which is a simple activation function. g is any non-linear function, such as sigmoid or hyperbolic tangent. Following the highway network's output, we pass the hidden embeddings to a dropout layer, which effectively reduces the number of hidden units by a fraction d, so $\boldsymbol{h}_{drop} \in \mathbb{R}^{k/d \times l}$, and a linear layer, which maps the $\boldsymbol{h}_{drop}$ to a smaller embedding space. We label this space $\boldsymbol{h} \in \mathbb{R}^{k/d \times f}$ (f being the dimensions of the feature space) for brevity.

3.4 Sequence Tagging Layer

In the sequence tagging layer, we use the combined embeddings to identify the most likely sequence of tags for the input sentence. With the aggregated combined hidden state $\boldsymbol{h}$, we have the information required to assign tags to the words of the input sentence. For this, we use conditional random fields (CRF). The traditional formulation of a CRF can be written, given a set of observations sequences $X = x_1, x_2, ..., x_k$ and sequence of labels $Y = y_1, y_2, ..., y_k$ as,

$$p(Y|X; W, b) = \frac{\prod_{i=1}^{k} \exp\left(y_{i-1}, y_i, X\right)}{\sum_{y' \in \mathcal{L}} \prod_{k}^{i=1} \exp\left(y'_{i-1}, y'_i, X\right)} \tag{10}$$

where $\mathcal{L}$ is the set of possible labels in the tagset.

Since the observation sequence in our formulation is essentially the output vector $\boldsymbol{h}$, we can simplify the above equation by performing softmax to score the likelihood of a label being assigned. Therefore, the probability distribution is computed as,

$$P(y_i = t|h_i) = \frac{\exp\left(h_i^T w_j + b_j\right)}{\sum_k \exp\left(h_i^T w_m + b_m\right)} \tag{11}$$

with $j, m \in \mathcal{L}$ as tag labels. We also compute the transition probability T of the label y_i being assigned to h_i given the labels of h_{i-1}. Therefore, the probability of the sequence of labels over the hidden states can be computed as:

$$Seq(Y, \boldsymbol{h}) = \sum_{i=1}^{k} P(y_i = t|h_i) + \sum_{i=1}^{k} T(y_i = t|y_{i-1} = t'); \; t, t' \in \mathcal{L} \tag{12}$$

Therefore the probability of that sequence Y computed above is calculated as:

$$p(Y|\boldsymbol{h}) = \frac{\exp\left(Seq(Y, \boldsymbol{h})\right)}{\sum_{y' \in \mathcal{L}} \exp\left(Seq(y', \boldsymbol{h})\right)} \quad (13)$$

4 Experimental Setup

In this section, we go over the various experiments, implementation details such as number of epochs, training time, datasets and the like. These are covered in detail for the replicability of our results, which are highlighted in section 5.

4.1 Datasets

To train and evaluate our model, we use the following datasets for each of the languages we work with multiple corpora, as our experiments span multiple languages.

1. The TempEval-3 TimeBank dataset was used for English (UzZaman et al., 2012). The corpus consists of 61,418 tokens for training and 6,756 event mentions.

2. For Spanish, we use the ModeS TimeBank (Modern Spanish TimeBank 1.0) (Nieto and Saurí, 2012) for training and testing. This was used in SemEval-2013 Task 1 Task B (UzZaman et al., 2013). The corpus consists of 57,977 tokens.

3. For Italian, we use Ita-TimeBank's ILC corpus (Caselli et al., 2011a) the Italian corpus annotated using ISO-TimeML rules for events and temporal information. The corpus consists of 68,000 tokens and 10,591 event mentions.

4. For French, we use the French TimeBank as it is the ISO-TimeML annotated reference corpus for event annotation tasks (Bittar et al., 2011). The corpus consists of 16,208 tokens and 2,100 event mentions.

5. For Hindi, we use the gold-standard corpus of Goud et al. (2019b), which consists of 810 event annotated news articles based on modified TimeML rules. The dataset has 242,201 tokens and 20,190 event mentions.

4.2 Model Implementation and Training Details

The datasets are annotated in the IOB format. At a word level, B represents the first token of an event, I represents all the other tokens of an event and O represents the tokens which are not a part of any event in the sentence. We train the model for 50 epochs, but the loss tends to stabilize at 25 to 35 epochs. We use a 40 dimensional character embedding, which we create ourselves, as mentioned in section 3.1. The CNN uses 40 filters with a window size of 3.

For our contextual word embeddings, we use fastText embeddings for English (Bojanowski et al., 2017) which are pretrained on commonCrawl and the Wikipedia corpus. FastText embeddings are also used for Hindi, French, Spanish and Italian word representations (Grave et al., 2018). The bi-LSTM trains on a fixed 300 hidden dimensions for all the bi-LSTMs in the architecture.

For the linear and dropout layers, the dropout is fixed to 0.3. The initial learning rate parameter is 0.015, which increases with a momentum of 0.9. On approaching the end of an epoch, the learning rate decays at a rate of 0.05. We train on a negative log-likelihood loss function

5 Results and Analysis

In this section, we analyze the results of the ALINED model, and compare them to the current state of the art systems for the various languages we train on. We also provide a rigorous error analysis of our system and methodology.

Since no single system has compared work in event detection across the five languages that we have chosen for the experiments here, we draw comparisons to the various systems that trained on the individual or group of languages that have been used. Table 1 ahows the direct comparison of results.

1. For English, we compare our system to the SemEval-2013 Task 1 Task B (UzZaman et al., 2013), detection of event extents. We compare our models' scores with those of the best performing models of SemEval-2013.

2. SemEval-2013 Task 1 Task B (UzZaman et al., 2013) performs the task of detecting event extents in Spanish texts. We compare our model performance to FSS-TimeEX and TipSemB-F, the best performing models in that task.

3. Caselli et al. (2011b) establishes the current state of the art for data driven models in temporal and event extent information in Italian.

Language	Model	Precision	Recall	F1-Score
English	ATT-1 (Jung and Stent, 2013)	81.44	80.67	81.05
	ATT-2 (Jung and Stent, 2013)	81.02	80.81	80.91
	ATT-3 (Jung and Stent, 2013)	**81.95**	75.57	78.63
	KUL (Kolomiyets and Moens, 2013)	80.69	77.99	79.32
	ALINED	78.79	**87.00**	**82.70**
Spanish	FSS-TimEX (Zavarella and Tanev, 2013)	89.80	42.40	57.60
	TIPSemB-F (UzZaman et al., 2013)	**91.70**	**86.00**	**88.80**
	ALINED	86.77	83.22	84.96
Italian	TIPSemIT_basic (Caselli et al., 2011b)	90.00	77.00	83.00
	TIPSemIT_FPC5 (Caselli et al., 2011b)	89.00	81.00	85.00
	TIPSemIT_FPC5Sem (Caselli et al., 2011b)	**91.00**	**83.00**	**87.00**
	ALINED	79.92	81.85	80.87
French	CRF-kNN (Arnulphy et al., 2015)	**87.00**	79.00	**83.00**
	Bittar (2009)	46.00	**82.00**	64.00
	ALINED	84.48	67.12	74.81
Hindi	**ALINED**	**78.22**	**76.08**	**77.13**

Table 1: Comparison of Model Performance

The system is a modification of the TipSem system. We compares our models to their reported scores. However, the corpus used in Caselli et al. (2011b) is the Ita-TimeBank which has been augmented with further annotations and resources, while our system uses just the Ita-TimeBank for event extraction.

4. For French, we did not find systems that did event extraction from the French TimeBank corpus. The existing literature either creates and evaluates on a modified corpus (Bittar, 2009) or provides annotations trained on the TimeML annotated data and tested on Fr-TempEval2) (Arnulphy et al., 2015). Therefore, we compare our performance to those, while also understanding that the comparison is not a strict metric. We hope to establish the scores here as baseline for further improvement over models in event detection in French.

5. To the best of our knowledge, there is no baseline system available for event detection in Hindi, therefore, we provide our model as the first performance metric in that direction.

In most comparisons, our models perform equally well or better than the current systems for each of the above languages. we do not annotate or augment any of our data sources for using this model, so the reference corpora are being trained and tested upon, which are mentioned in section 4.1.

The calculation of the metrics of comparison, precision, recall and accuracy are calculated as follows:

$$precision = tp/(tp + fp)$$
$$recall = tp/(tp + fn)$$
$$f - measure = 2 * (P * R)/(P + R)$$

where tp is a true positive, where the part of the extent identified in the system output is the same as the expected output, fp is a false positive, where the token identified as part of the extent by the system is not a part of the expect output, and fn is a false negative, where a token not identified as a part of the extent by the system output, is a part of the expected output.

We note a lower precision score in case of English and Spanish, as the number of false positives are slightly higher. We attribute this difference to the fact that due to the combination of sub-word level features, the model seems to sometimes "spill over" the boundary of single word or nominal. However, higher recall implies that there

are fewer false negatives, meaning the model more accurately identifies those words which are in the event span. More labeled data would be very useful in learning the span boundaries, especially for nominal events, as the network would have more samples to learn the variations in the methods of event representation.

For English, surprisingly, we see that an increase in the F1-scores. We attribute this to a combination of factors, including well defined verbal affixes which are attributed to events, and effective weighted combination of character and word embeddings.

For Italian, we train and test solely on the Ita-TimeBank, whereas the current state of the art system trained on an augmented Ita-TimeBank (Caselli et al., 2011b), which was enriched with more labeled data. Similarly, in French, we use the established French TimeBank, while experiments in French so far have been on self-annotated (Arnulphy et al., 2015) or TimeML corpora (Bittar, 2009). Since these repositories of augmented data were not available to us at the time of writing this paper, the values reflect the same. However, it is to be noted that our system does provide an accuracy that is close to the currently reported state-of-the-art even in the absence of language specific features, explaining the fact that sub-word information is necessary for event detection in Italian and French as well.

For Hindi, our architecture provides a good baseline. However, the training data consists of far too many words that are out of vocabulary, which is a major issue in working with word embeddings. While the concatenation of sub-word information mitigates this, a system focused on a better representation of out of vocabulary words would significantly help the network. However, this required a larger labeled corpus as well, which makes this a challenge as Hindi is a low-resource language in terms of corpora for event detection and extraction.

6 Conclusion

In this paper, we show the development of ALINED, a language invariant neural sequence tagging architecture for event detection in five different languages, namely, English, Spanish, Italian, French and Hindi. We develop insight into the use of sub-word level information and combining it effectively. with the lexical and syntactic information.

For our training and testing, we use only established corpora, which have not been augmented or changed in any way. We perform almost at par or better then the current state of the art in all the languages we train in. We establish a new best F-score for event extraction in English. We also establish the baseline for training and testing on the French TimeBank and for event extraction as a task in Hindi.

Our model has been thoroughly error-analyzed, which we have explained based on the comparison of system output and expected tags. Given the nature of our results, we aim to establish the importance of sub-word level information in event detection. Further work in this task could be done by providing augmented reference corpora, so that problems based on lack of labeled data do not limit further research in this topic. This could also be tackled by effectively introducing transfer learning to neural event detection, where the model learns the representation of events irrespective of language, while accounting for sub-word, lexical and structural information.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Jun Araki. 2018. *Extraction of Event Structures from Text*. Ph.D. thesis, Ph. D. thesis, Carnegie Mellon University.

Béatrice Arnulphy, Vincent Claveau, Xavier Tannier, and Anne Vilnat. 2015. Supervised machine learning techniques to detect timeml events in french and english. In *International Conference on Applications of Natural Language to Information Systems*, pages 19–32. Springer.

Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. 2013. Advances in optimizing recurrent networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8624–8628. IEEE.

André Bittar. 2009. Annotation of events and temporal expressions in french texts. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 48–51. Association for Computational Linguistics.

André Bittar, Pascal Amsili, Pascal Denis, and Laurence Danlos. 2011. French timebank: an iso-timeml annotated reference corpus. In *Proceedings of the 49th Annual Meeting of the Association for*

Computational Linguistics: Human Language Technologies: short papers-Volume 2, pages 130–134. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tommaso Caselli, Valentina Bartalesi Lenzi, Rachele Sprugnoli, Emanuele Pianta, and Irina Prodanof. 2011a. Annotating events, temporal expressions and relations in italian: the it-timeml experience for the ita-timebank. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 143–151. Association for Computational Linguistics.

Tommaso Caselli, Hector Llorens, Borja Navarro-Colorado, and Estela Saquete. 2011b. Data-driven approach using semantics for recognizing and classifying timeml events in italian. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 533–538.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 167–176.

Hai Leong Chieu and Yoong Keok Lee. 2004. Query based event extraction along a timeline. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 425–432. ACM.

Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. 2017. Neuroner: an easy-to-use program for named-entity recognition based on neural networks. *EMNLP 2017*, page 97.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, page 1.

Xiaocheng Feng, Bing Qin, and Ting Liu. 2018. A language-independent neural network for event detection. *Science China Information Sciences*, 61(9):092106.

Reza Ghaeini, Xiaoli Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 369–373.

Goran Glavaš and Jan Šnajder. 2014. Event graphs for information retrieval and multi-document summarization. *Expert systems with applications*, 41(15):6904–6916.

Jaipal Goud, Pranav Goel, Allen J. Antony, and Manish Shrivastava. 2019a. Leveraging multilingual resources for open-domain event detection. In *Proceedings 15th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, pages 76–82.

Jaipal Goud, Pranav Goel, Alok Debnath, Suhan Prabhu, and Manish Shrivastava. 2019b. A semantico-syntactic approach to event-mention detection and extraction in hindi. In *Proceedings 15th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, pages 63–75.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Hyuckchul Jung and Amanda Stent. 2013. Att1: Temporal annotation using big windows and rich syntactic and semantic features. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 20–24.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Oleksandr Kolomiyets and Marie-Francine Moens. 2013. Kul: data-driven approach to temporal parsing of newswire articles. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 83–87.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 73–82.

Dongyun Liang, Weiran Xu, and Yinge Zhao. 2017. Combining word-level and character-level representations for relation classification of informal text. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 43–47.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1064–1074.

Teruko Mitamura, Zhengzhong Liu, and Eduard H Hovy. 2015. Overview of tac kbp 2015 event nugget track. In *TAC*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 365–371.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 886–891.

Marta Guerrero Nieto and Roser Saurí. 2012. Modes timebank 1.0. *Linguistic Data Consortium (LDC), Philadelphia, PA, USA*.

Tim O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer event description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*, pages 47–56.

James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. 2010. Iso-timeml: An international standard for semantic annotation. In *LREC*, volume 10, pages 394–397.

James Pustejovsky, Jessica Littman, Roser Saurí, and Marc Verhagen. 2006. Timebank 1.2 documentation. *Event London, no. April*, pages 6–11.

Roser Saurı. 2010. Annotating temporal relations in catalan and spanish timeml annotation guidelines. Technical report, Technical Report BM 2010-04, Barcelona Media.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2012. Tempeval-3: Evaluating events, time expressions, and temporal relations. *arXiv preprint arXiv:1206.5333*.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 1–9.

Y Wen, R Luo, J Wang, et al. 2016. Learning text representation using recurrent convolutional neural network with highway layers. In *Neu-IR: The SIGIR 2016 Workshop on Neural Information Retrieval*, volume 2016. Association for Computing Machinery (ACM).

Yang Xiang, Xiaoqiang Zhou, Qingcai Chen, Zhihui Zheng, Buzhou Tang, Xiaolong Wang, and Yang Qin. 2016. Incorporating label dependency for answer quality tagging in community question answering via cnn-lstm-crf. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1231–1241.

Vanni Zavarella and Hristo Tanev. 2013. Fss-timex for tempeval-3: Extracting temporal information from text. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 58–63.

Event Centric Entity Linking for Hindi News Articles: A Knowledge Graph Based Approach

Pranav Goel, Suhan Prabhu, Alok Debnath and **Manish Shrivastava**
International Institute of Information Technology
Hyderabad, Telangana, India
{pranav.goel, suhan.prabhuk, alok.debnath }@research.iiit.ac.in
m.shrivastava@iiit.ac.in

Abstract

We describe the development of a knowledge graph from an event annotated corpus by presenting a pipeline that identifies and extracts the relations between entities and events from Hindi news articles. Due to the semantic implications of argument identification for events in Hindi, we use a combined syntactic argument and semantic role identification methodology. To the best of our knowledge, no other architecture exists for this purpose. The extracted combined role information is incorporated in a knowledge graph that can be queried via subgraph extraction for basic questions. The architectures presented in this paper can be used for participant extraction and event-entity linking in most Indo-Aryan languages, due to similar syntactic and semantic properties of event arguments.

1 Introduction

Events are defined as situations that happen or occur (Saurí et al., 2006). Events therefore involve participating entities, sometimes referred to as event arguments (Ji and Grishman, 2008). The extraction of role information of entities participating in events is a fast-evolving area of research in information retrieval as well as subfields of NLP such as question answering and summarization (Lin and Liang, 2008). This paper handles the challenge of participant detection and labeling in Hindi, using syntactic measures such as dependency parsing and semantic measures such as verb frame comparisons and semantic role labeling. Using the entities extracted from the text and their relation to the event, a knowledge graph is generated, which can then be queried for basic questions.

In Hindi NLP, the representation, identification and extraction of events is a fairly new concept. Event extraction from twitter data (Kuila and Sarkar, 2017) and in news data (Ramrakhiyani and Majumder, 2013; Goud et al., 2019) are still developing areas of research. However, extensive work has been done on argument structure for Hindi verbs, therefore the syntactic analysis of verbal events has been a topic of sufficient inquiry (Butt, 2010). On the other hand, nominal events, while not studied under that paradigm, have been referenced in entity linking in NER research (Athavale et al., 2016).

This paper, given the definition of events in Hindi (Goud et al., 2019), identifies the arguments of these events. We employ a syntactico-semantic approach of entity identification by using dependency parsing to determine the syntactic roles of the arguments and their dependency length from the event mention (Gulordava et al., 2015), and a semantic role labeler which is used to determine the semantic case or functions of the participating entities (Carreras and Màrquez, 2005). For verbal events, verb frame data has also been used for verifying the arguments. This information is constructed as a knowledge graph, a query graph (Yih et al., 2015) of which can then be used for question answering.

2 Related Work

Entity or participant extraction is a vital subdomain of event detection and related information extraction tasks. The ACE project (Doddington et al., 2004) and many of the relevant event extraction tasks that followed it had entity detection and tracking as one of the main components for event detection and extraction systems (Ahn, 2006). ACE also provided twenty-four different types of relations between entities. Hong et al. (2011) establishes a mechanism of using entity links in order to more accurately detect event mentions, by associating some entities as event par-

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 45–55
Hyderabad, India, December 2019.

ticipants or arguments. Joint extraction of event and entity mentions has been attempted (Yang and Mitchell, 2016) by learning intra-event structures and possible forms of entity relations to events.

Named entity recognition has been another broader form of approach to entity identification and linking. Entity mention detection and tracking its use in the corpus (Xu et al., 2017) is considered the most fundamental method in this approach. Yamada et al. (2015) approaches the problem of named entity recognition from the perspective of entity linking. Hybrid joint approaches to participant extraction and linking (Plu et al., 2015) have been treated as an extension of this problem, and the OKE 2017 task (Plu et al., 2017) performed participant extraction and linking for ontology enrichment. Florian et al. (2004) and Lin et al. (2016) perform cross lingual entity linking over an enriched knowledge base, one of the languages being Hindi. These approaches are important for understanding and disambiguating the links between nominal events and their participants.

Argument analysis for verbs in Hindi has been a well-researched topic, as mentioned above. Palmer et al. (2009) studies the computational properties of verbal predicates from a dependency annotation perspective, while Vaidya et al. (2016) and Vaidya et al. (2019) focuses on the syntactic argument structure of light verbs in Hindi. Light verbs are one of the syntactic constructions observed in representation of eventive verbs. Compound verb detection (Chakrabarti et al., 2008), complex predicate detection (Mukerjee et al., 2006) and argument identification in complex predicates (Montaut, 2016) can be modeled together in syntactic argument detection for verbal events. The study of noun incorporation in verb complexes (Dayal, 2015) provide a semantic perspective of argument structure and event participation. Syntactically, two major concerns of verb argument analysis are verb phrase ellipsis and complex predicate analysis (Manetta, 2018b,a).

Knowledge graphs are extensively used in semantic information retrieval and has numerous other applications cross language document retrieval (Franco-Salvador et al., 2014), cross lingual plagiarism detection Franco-Salvador et al. (2016), question answering (Indurthi et al., 2017) and summarization (Zheng et al., 2016).

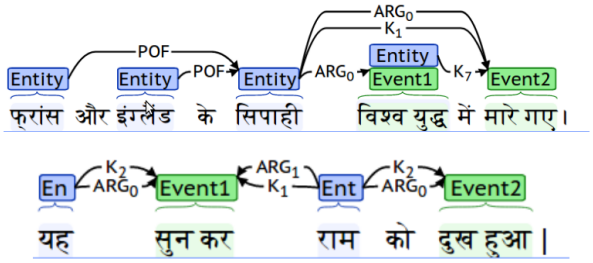


Figure 1: Example of a tagged pair of sentences. The event indexes are intra-sentence.

Overall statistics	
Number of Articles	810
Total Number of Sentences	13949
Total Number of Events	20190
Nominal Events	1841
Verbal Events	18349
Total Number of Entities	41847
Average per sentence	
Length (Words)	18
Number of Entities	3
Number of Events	1.48
Number of Entities per Event	2.08
Most Common Relation	
Entity - Nominal Event	(ARG0)
Entity - Verbal Event	(K1, ARG0)
Inter-Annotator Statistics	
Participant Identification	0.86
Syntactic Role Identification	0.89
Semantic Role Identification	0.79
Coreferent Mention Identification	0.91

Table 1: Dataset and Annotation Statistics

3 Dataset and Annotation Specifications

We use a gold-standard corpus of 810 news articles of Goud et al. (2019), and annotate it for entities and their relations with the events. The entity-event relations are annotated based on a syntactic as well as a semantic role. The syntactic role is simply a dependency label (Tandon et al., 2016), while the semantic labels are provided according to Hindi and Urdu PropBank labels (Bhatt et al., 2009).

The annotated sentence shown in Figure 1 is:

frans	*aur*	*england*	*ke*	*sipahi*
France	and	England	of	soldiers
vishva	*yudha*	*mein*	*maare*	*gaye.*
World	War	in	killed	got.

ye	*sun*	*kar*	*ram*	*ko*
This	listen	to	Ram	(acc)
bohot	*dukh*	*hua*		
much	sad	happen		

Table 1 presents some of the basic statistics of the annotated data. The data has been annotated by four annotators who are proficient in Hindi and are students of linguistics using the BRAT annotation tool (Stenetorp et al., 2012) for annotating and providing labels. The inter-annotator agreement was measured by a strict match Cohen's Kappa Score (Cohen, 1960).

The dataset is then annotated further for ease of semantic role extraction. All event mentions are indexed, and if two event mentions are coreferent, they are given the same index. In case of entities, only entities with coreferent mentions are indexed. Coreferent entity mentions are given the same index. For this task, inter-annotator agreement was calculated on four different measures, identifying the participant, correct syntactic role, correct semantic role and correct coreferent mention identification. For the purpose of coreference, the entities and events are treated the same.

4 Identifying Entity Participation in Events

In this section, we look at the pipeline for the extraction of entities as arguments of events. We define here a nominal event as an event which has the event nugget (the core of the event) is a noun. Similarly, a verbal event is an event which has the event nugget which is a verb.

As discussed before, discerning entity participation in events is a syntacto-semantic problem, and therefore our solution (refer to Figure 2) has both syntactic and semantic components. Note that a IOB (inside-outside-begin) tagged event mentioned corpus is the input to the pipeline. The outputs from each module and the final pipeline are formatted to be used in the form of a knowledge graph, which is detailed in Section 5.

4.1 Syntactic Participation Detection

The syntactic components are essential preprocessing tasks such as POS tagging and dependency parsing. The dependency parse provides the syntactic role information. Particularly for verbal events, the parse also provides the distance from the event mention, which is essential in order to determine participation in sentences with multiple events. This participation is then verified using verb frame data.[1]

The procedure for syntactic participation detection is as follows:

1. A Hindi POS tagger (Shrivastava and Bhattacharyya, 2008) is used to identify the part of speech of all the lexical items in the sentence. Since most event arguments are nominal or pronominal, the relevant words are extracted. The POS tagged text is then provided as input to the next phase.

2. The text is then parsed using a dependency parser (Palmer et al., 2009). The dependency labels from the root are considered most important, since the nouns and pronouns directly associated with the verb are most likely to be the arguments of the event.

3. The *karaka* and *sambandh* edge labels, which are provided by the dependency parser, are extracted. The *karaka* edge labels provide the case of the noun and its role with respect to the verb, while the *sambandh* edge label mark the genitive relation between two nouns. If either one of the nouns is eventive, the relation given to it is the relevant *karaka* relation. If both the nouns are entities, then they are not linked. The relation between two entity nouns by a *sambandh* (genitive) case marker is not marked in the graph directly. Instead, genitive chains are constructed after extraction of the entity, using the dependency tree.

The dependency parser provides syntactic role information and the distance of the extracted words from the verbs in the sentence. For sentences with relative or subordinate clauses, as well as multiple events, this feature is used to determine which event is linked to which entity. The genitive *sambandh* relations are retained irrespective of the eventiveness of the nouns for the purpose of identifying the primary participant in an event in case of a long genitive chain.

[1] While we define entities by participation similar to ACE (Doddington et al., 2004), the definition of event (Goud et al., 2019) allows for multiple events in a single sentence.

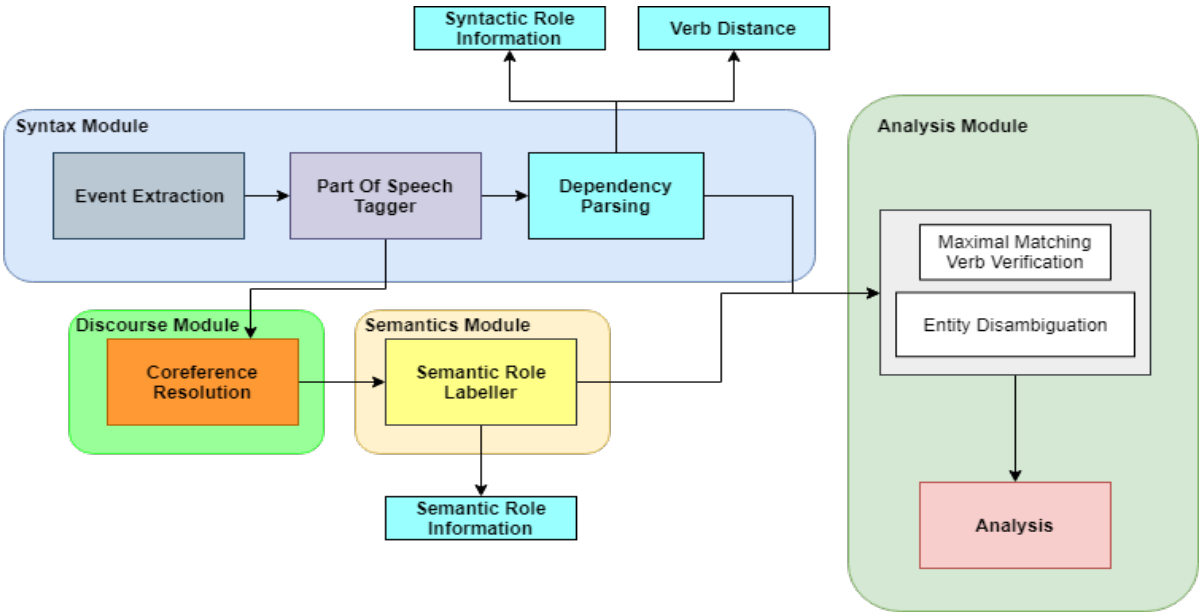


Figure 2: Pipeline for Participant Extraction

4.2 Semantic and Discourse Relation Extraction

The semantic role of the arguments to an event are extracted by the semantic role labeler (SRL) for Indian languages (Anwar and Sharma, 2016). The SRL uses POS tagged text as an input and provides the semantic role of the nouns and adverbs in the sentence. For the purpose of participant extraction, the adverbs are ignored.

However, before the semantic role extraction can be done, event coreference, entity coreference and anaphora resolution are permormed, in order to determine the possible overlap of event mentions (multiple event mentions for the same event) (Chen et al., 2009).

- **Event coreference** is taken care of by indexing the event. All event mentions in the annotated input are indexed by a numerical subscript. Corefernent events have similar event triggers and overlapping argument structures (Lu et al., 2016), which are crucial features in the annotation of these events. The indices of coreferent event mentions are the same, which indicates that they share their arguments.

- **Entity coreference** is taken care of determining the role that the entity performs in the event. This is one of the primary entity based features used for entity coreference (Clark and Manning, 2015). In the corpus, the entities are partially indexed, that is that only coreferent entities are indexed.

Both anaphora and event coreference are done automatically using a combination of role extraction and verb relations as mentioned above, as well as using pretrained models (Devi et al., 2014) and manual editing of the output.

After this, if a noun also happens to be an event, the dependency relation between it and a verbal event in the sentence (if any) is retained, while the semantic relation is removed. Event-event relations are beyond the scope of this paper, and for the sake of simplicity, it is assumed that events can not be arguments to other events. Retaining the dependency information, however, as it is a feature used in entity disambiguation if an entity happens to participate in a nominal and a verbal event. As with the dependency parse, the semantic relations between two nouns is retained regardless of their eventiveness, as the semantic relation acts as a verification for the detected primary participant.

4.3 Role Analysis and Verification

In order to accurately determine the roles assigned by the two modules above, our pipeline is equipped with an analysis module. In line with the Paninian tradition, we use the notion of *yogyata* (capability) (Kulkarni et al., 2010) to verify whether an event can take the types and roles of the arguments that have been assigned to it. The output of this system are then analyzed as tem-

plates of entity-event relations, which are used to create the knowledge graph.

Verbal events are analyzed using verb frame data (Soni et al., 2013). The verb frame data provides the possible *karaka* relations which can be used to determine the mandatory and optional syntactic expectancy of the verb in different senses. A maximal matching algorithm (Algorithm 1) is used across all senses, and the sense with *all* mandatory and the maximum number of optional *karaka* arguments is chosen as the sense of that verb.

Algorithm 1 Maximal matching Verb Verification

procedure MAXMATCHVERB
  VFD ← Verb Frame Data
  $V \leftarrow$ Verbal Event
  $part \leftarrow$ list *[(Parent, Participant, Role)]*
  $max\_all \leftarrow 0$
  $max \leftarrow -1$
  for $verb$ **in** VFD **do**
    for $sense$ **in** $verb$ **do**
      if $(V = verb)$ and $(part[2] = sense)$ **then**
        $max = max + 1$
    if $(max\_all < max)$ **then**
      $max\_all \leftarrow max$
  return max_all

Algorithm 2 Entity Disambiguation

procedure ENTITYDISAMBIGUATION
  $N \leftarrow NominalEvent$
  $E \leftarrow EntityList$
  if N **in** E **then**
    Remove N from E
  $V \leftarrow$ Closest verbal event from N ▷ Word distance or tree distance
  Add ARG_0, ARG_1 of V to E
  $VEList \leftarrow$ list of all verbal events
  for $VE \in VEList$ **do**
    if $ARG2_{LOC} \in VE$ **then**
      Add $ARG2_{LOC}$ of VE to E
    if $ARG2_{GOL} \in VE$ **then**
      Add $ARG2_{GOL}$ of VE to E
    if $ARG_{SOU} \in VE$ **then**
      Add ARG_{SOU} of VE to E
  if ARG_{TMP} exists **then**
    if $ARG_{TMP} \notin arg(VE)$ for $VE \in VEList$ **then**
      Add ARG_{TMP} to E

Nominal entity participant identification follows two steps, jointly referred to as entity disambiguation. First, we use a naive coreference resolution using a feature set similar to Dakwale et al. (2013)'s rule based implementation, for entities and events. The syntactic roles of significance are *sambandh* relations. Some of the design choices in Lee et al. (2012), including features such as number of coreferent arguments and argument roles are crucial to determining participation, as shown in Algorithm 2.

Finally, we analyze and resolve co-participation ambiguities. For sentences with multiple events, it is necessary to verify whether all the entities necessarily participate in, or are modified by, the attributed events. In verbal events, maximal matching is done on the entities syntactically closest to it, which performs well in default word order. For entities linked to both nominal and verbal events, semantic role information is considered. Nominal events characteristically only take agentive, thematic and locative arguments over the verbal predicate (Gerber and Chai, 2012), while only those temporal arguments are taken which are not attributed to the verbal event.

After completing this analysis, the output of the pipeline is condensed and reformatted as inputs to a knowledge graph.

5 Entity-Event Knowledge Graph

Knowledge graphs have been widely used in information retrieval, since their adoption in popular search engines. However, knowledge graphs can be constructed for document wide, corpus wide or domain wide extraction of information as well. In this section, we show the development of an event-centric entity linked knowledge graph.

(Rospocher et al., 2016) defines an event centric knowledge graph as a knowledge graph in which all information is related to events through which the knowledge in the graph obtains a temporal dimension. Knowledge graphs are useful for the representation of semantic information in the edge labels or in the attributes of the nodes itself. Document wide knowledge graphs can be queried by limiting the search space based on the query. This method of creating a query graph allows for an inference chain for the related nodes (Yih et al., 2015).

Question word	Gloss	Category	Role
kis + case marker[2]	who	Entity	-
kaun	who	Entity	-
kahAn	where	Entity	Location or Source
kab	when	Entity	(Time)
kyun	why	Event or Entity	(Goal)

Table 2: Question Words and Answer Types

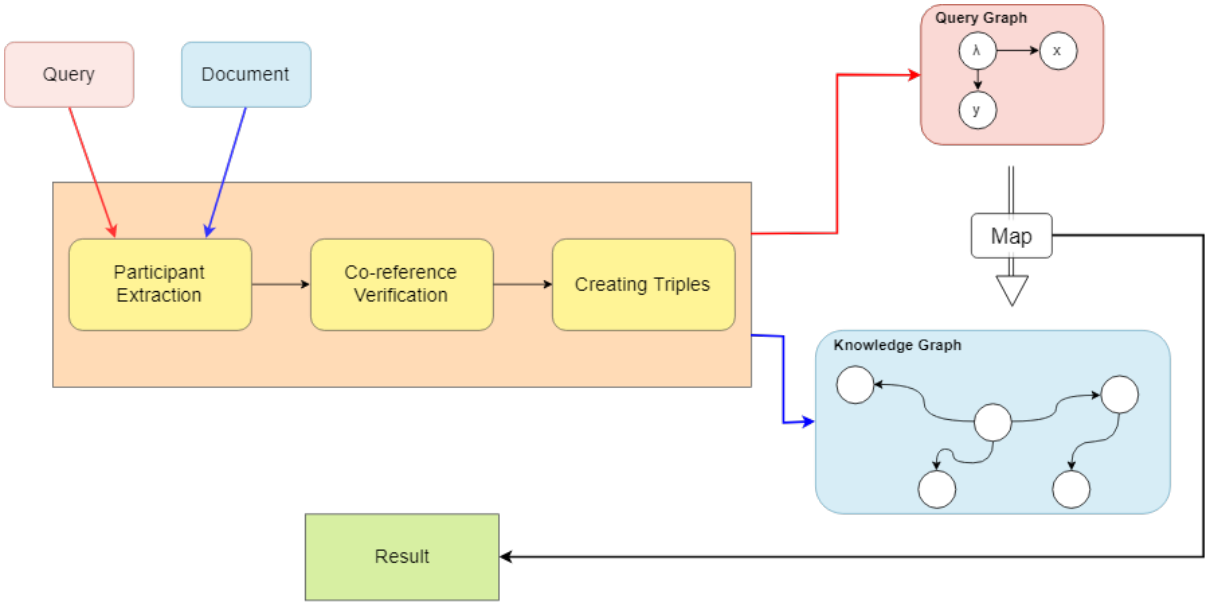


Figure 3: Pipeline for Knowledge Graph and Query Graph Creation. "Participant Extraction" refers to the Pipeline in Fig. 1

5.1 Developing the Knowledge Graph

In order to develop a knowledge graph, we must determine the relevant nodes and edges. We choose to consider events and entities as nodes, and the relations between them as the edges. The relations between them, as mentioned before, are both syntactic and semantic. We show the development of the knowledge graph and handling queries in Figure 3

Creating triples As with most knowledge graph based representations, the first step is to extract the necessary triples that constitute the graph. The data after being passed through the entity detection and linking pipeline, has to be reformatted into $(e,(n,m),v_i)$ triples, where $e \in E$, the set of all entities, $(n,m) \in (N \cup \{\phi\}, M)$, where N is the set of all syntactic roles and M is the set of all semantic roles, and $v_i \in V$, the set of all events in the document, indexed. If there is no syntactic role of the entity in an event, as is common with nominal events, the syntactic role given to it is ϕ.

We also construct specific *genitive triples*, defined as (e_i, n, e_j) where $e_i, e_j \in E$, the set of all entities and n is always given a POF relation. These links are useful when constructing entity links and chains. The genitive triples are not used directly in the knowledge graph. Instead, the constructed genitive triples (since they are directly extracted from the dependency graph) are used for generating an answer for a query. These are maintained primarily for efficiency in generating an answer for the query.

Handling Event Coreference After triple creation, event coreference has to be handled. Coreference is handled in semantic role extraction. Events are indexed by the occurrence of their first mention in the text. A relation has to be created between the entities of events with the same index. Note, however, that because of the temporal nature of events, entities that are linked to a later mention of a coreferent event are not linked to the first. In our approach, the first mention of an event is considered its *primary mention*, for the purpose of creating the knowledge graph. All

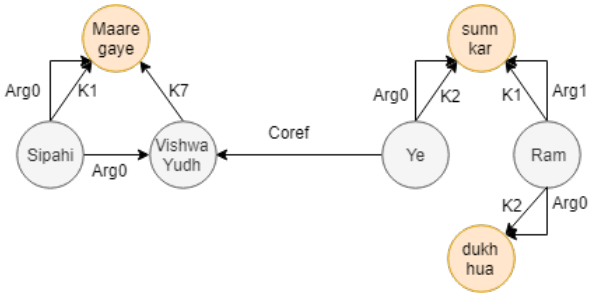


Figure 4: Knowledge graph representation of fig. 1

other mentions are *secondary mentions*, which are ordered through the document in their order of occurrence. The entities participating in the primary event mention are considered participants to all the secondary mentions, while the arguments of the first secondary mention are arguments only to itself and the remaining secondary mentions and so on. Therefore, for each new event mention of the same event, new triples are made which account for the participation of that event from all previous event mentions.

Handling Entity Coreference In the case of entity coreference and anaphora, entities with multiple mentions are already indexed, and therefore, all the entity mentions are considered the same entity, and if an entity mention participates in an event, all other entity mentions participate in it as well. Therefore using the index values of the entities, a coreference chain can be formed that defers all entity mentions to the *primary entity mention*, which is the first mention of an entity. This choice also makes query graph formation easier. Therefore, all triples where the entity is indexed are replaced with the primary mention of the entity.

Figure 4 shows the knowledge graph of a snippet of the sentences from figure 1. Events and entities are both nodes, as mentioned; we use colors to distinguish between them.

5.2 Querying the Knowledge Graph

Once the knowledge graph has been created, it can be used for other downstream tasks and applications. One of the major applications is question-answering. Recent approaches to open domain question-answering systems over graph databases like Freebase (Bollacker et al., 2008) follow a semantic parsing approach (Yao and Van Durme, 2014). Our approach for querying the entity-event knowledge graph is similar to Yih et al. (2015)'s approach. We generate a query graph of the question and perform predicate matching over the λ-expression corresponding to the query graph after exhausting all possible inference chains.

We are first tasked with the annotation of events and entities in the question. Event annotation is done by the the methods described in (Goud et al., 2019), and are not discussed in the scope of this paper. But given an event annotated event sentence, we first identify the entities and the *question entity*, which is the interrogative pronoun. The basic pipeline for entity recognition in the document is also followed for the question. In the analysis phase, the question pronoun is marked. We map the question pronoun to the type of response expected, that is, either an entity or an event. Using this information, the query graph is created, from which a λ-expression is extracted.

Carrying the example from section 3, a factoid question based on the sentences (sentences in 1, graph in 4) could be:

yudh	*mein*	*kaun*	*maare*	*gaye*?
War	in	who	killed	got?

As mentioned, we first parse and analyze the question, as has been done before in section 4. The dependency parse provides us with which word is the question word. We also use a specific morphanalysis module to extract syntactic role (*karaka*) information.

From this, we construct the λ-expression $\lambda x.\exists y : \texttt{entity}(x, y) \wedge \texttt{Arg0}(y, \textit{maare gaye}) \wedge \texttt{k1}(y, \textit{maare gaye}) \wedge \texttt{Arg0}(y, \textit{yudh})$. The first of the relations (`entity`), can be determined based on the question word's role in the sentence. For the purpose of factoid questions on our dataset, only the question *kyun* (why) is considered to have an answer which is tagged `event`. Table 2 is the simple mapping from question to query. In the cases where a question can have multiple types of answers, the largest number of overlapping words is considered the disambiguating heuristic. Question words such as *kaise* (how) and *kyA* (what) are not accounted for, as not all formats of the question are factoid in nature. Therefore, using the lambda-expression, we can construct a query graph, which can easily be mapped onto the knowledge graph, and the y is the answer to the query, while $\lambda(x)$ ascertains whether the answer is of the correct type (event or entity).

Entity Detected	Overlap	Label
Overall	86.4%	84.1%
Nominal Events	64.1%	71.7%
Verbal Events	93.7%	89.4%

Table 3: Average Accuracy for Participant Extraction

6 Analysis and Results

In this section, we look into the two pipelines which have been developed for constructing a basic knowledge graph from an event annotated corpus, and the type of queries it can handle. We provide both a qualitative and quantitative analysis of the results of the pipelines. We also provide a thorough analysis of errors.

6.1 Participant Extraction Pipeline

The participant extraction pipeline (figure 2) has multiple interdependent components, such as the event annotated corpus, the POS tagger and dependency parser, the coreference resolution module and the semantic role labeler. Based on the annotated data, we find that the pipeline accurately detects the presence of 86.4% of the participants of the events for each event on an average. Table 3 shows the percentage of average complete overlap and the accuracy of label detection. Note that only complete overlap of the entity span is considered as the output and the label is considered accurate if all the roles have been correctly identified.

The relative drop in accuracy for nominal events is due two primary reasons, first that there are no syntactic features for the detection of participants in nominal events and secondly coreference of nominal events as entities. We notice that a coreferent event mention can act as an entity, but still hold eventive characteristics, which has not been handled in our pipeline. Furthermore, due to case marker overloading (Bharati et al., 2002) in Hindi, the accurate detection of labels is affected.

6.2 Knowledge Graph and Queries

In the creation of the knowledge and query graphs, illustrated in figure 3, we see that the errors of the participant extraction pipeline mentioned above will propagate forward, causing the knowledge graph to be an ill-representation of the document. As mentioned above, the characteristic error arose from coreference mishandling, and therefore, the coreference validation module accounts for the assigning the eventive nature of the coreferent event mentions which act as entities.

We qualitatively analyze the knowledge graph and the pipeline by using simple queries in order to verify the creation of the graph and the associated nodes and edges. The queries, as shown in Figure 3, also pass through the same pipeline, and we verify the knowledge graph based on the accuracy of the response to the query. Since the λ expressions are constructed based on simple rules based on the nature of Hindi question words, we could only qualitatively analyze the graph on simple queries with single query results.

Queries of the form '*kis*' + case marker or *kaun* provide a valid response to the query. However, for sentences with multiple events, queries provide incorrect results in some cases. This is partly because of entity sharing, which is that an entity is associated to multiple events if they are subevents. Since the relations between events have not been handled yet, and is beyond the scope of this paper.

7 Conclusion

In this paper, we attempted to determine a method of identifying the participants of each event in an event-annotated corpus, given the syntactic and semantic role of each noun and verb in a sentence. We used a distinct pipeline of interacting tools which provided various levels of syntactic and semantic information, which were then combined and analyzed. We have presented the two major algorithms; one for identifying the sense of the verb being used (based on the available mandatory arguments and the maximum match of optional arguments), and the other for determining the participants of a nominal event. We have also presented the development of a queryable knowledge graph on the basis of the events and entities extracted, that use the role information as edge labels. With this work, we hope to develop a more robust representation of events and entities, which can be enriched with developments in event classification and event relation extraction in Hindi. Most importantly, the pipeline and algorithms developed in this paper are language agnostic, which we hope will spur research into developing information rich representations of event and participation information in other languages as well.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and*

Reasoning about Time and Events, pages 1–8.

Maaz Anwar and Dipti Sharma. 2016. Towards building semantic role labeler for indian languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Vinayak Athavale, Shreenivas Bharadwaj, Monik Pamecha, Ameya Prabhu, and Manish Shrivastava. 2016. Towards deep learning in hindi ner: An approach to tackle the labelled data scarcity. In *13th International Conference on Natural Language Processing*, page 154.

Akshar Bharati, Rajeev Sangal, Vineet Chaitanya, Amba Kulkarni, Dipti Misra Sharma, and KV Ramakrishnamacharyulu. 2002. Anncorra: building tree-banks in indian languages. In *COLING-02: The 3rd Workshop on Asian Language Resources and International Standardization*.

Rajesh Bhatt, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Sharma, and Fei Xia. 2009. A multi-representational and multi-layered treebank for hindi/urdu. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pages 186–189.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Miriam Butt. 2010. The light verb jungle: Still hacking away. *Complex predicates in cross-linguistic perspective*, pages 48–78.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the ninth conference on computational natural language learning (CoNLL-2005)*, pages 152–164.

Debasri Chakrabarti, Hemang Mandalia, Ritwik Priya, Vaijayanthi Sarma, and Pushpak Bhattacharyya. 2008. Hindi compound verbs and their automatic extraction. *Coling 2008: Companion volume: Posters*, pages 27–30.

Zheng Chen, Heng Ji, and Robert Haralick. 2009. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *Proceedings of the workshop on events in emerging text types*, pages 17–22. Association for Computational Linguistics.

Kevin Clark and Christopher D Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1405–1415.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Praveen Dakwale, Vandan Mujadia, and Dipti M Sharma. 2013. A hybrid approach for anaphora resolution in hindi. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 977–981.

Veneeta Dayal. 2015. Incorporation: Morpho-syntactic vs. semantic considerations. In *The syntax and semantics of pseudo-incorporation*, pages 47–87. BRILL.

Sobha Lalitha Devi, Vijay Sundar Ram, and Pattabhi RK Rao. 2014. A generic anaphora resolution engine for indian languages. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1824–1833.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1.

Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, H Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. Technical report, IBM THOMAS J WATSON RESEARCH CENTER YORKTOWN HEIGHTS NY.

Marc Franco-Salvador, Paolo Rosso, and Manuel Montes-y Gómez. 2016. A systematic study of knowledge graph analysis for cross-language plagiarism detection. *Information Processing & Management*, 52(4):550–570.

Marc Franco-Salvador, Paolo Rosso, and Roberto Navigli. 2014. A knowledge-based representation for cross-language document retrieval and categorization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 414–423.

Matthew Gerber and Joyce Y Chai. 2012. Semantic role labeling of implicit arguments for nominal predicates. *Computational Linguistics*, 38(4):755–798.

Jaipal Goud, Pranav Goel, Alok Debnath, Suhan Prabhu, and Manish Shrivastava. 2019. A semantico-syntactic approach to event-mention detection and extraction in hindi. In *Proceedings 15th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, pages 63–76.

Kristina Gulordava, Paola Merlo, and Benoit Crabbé. 2015. Dependency length minimisation effects in short spans: a large-scale analysis of adjective placement in complex noun phrases. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 477–482.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1127–1136. Association for Computational Linguistics.

Sathish Reddy Indurthi, Dinesh Raghu, Mitesh M Khapra, and Sachindra Joshi. 2017. Generating natural language question-answer pairs from a knowledge graph using a rnn based question generation model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 376–385.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. *Proceedings of ACL-08: HLT*, pages 254–262.

Alapan Kuila and Sudeshna Sarkar. 2017. An event extraction system via neural networks. In *FIRE (Working Notes)*, pages 136–139.

Amba Kulkarni, Sheetal Pokar, and Devanand Shukl. 2010. Designing a constraint based parser for sanskrit. In *International Sanskrit Computational Linguistics Symposium*, pages 70–90. Springer.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500. Association for Computational Linguistics.

Fu-ren Lin and Chia-Hao Liang. 2008. Storyline-based summarization for news topic retrospection. *Decision Support Systems*, 45(3):473–490.

Ying Lin, Xiaoman Pan, Aliya Deri, Heng Ji, and Kevin Knight. 2016. Leveraging entity linking and related language projection to improve name transliteration. In *Proceedings of the Sixth Named Entity Workshop*, pages 1–10.

Jing Lu, Deepak Venugopal, Vibhav Gogate, and Vincent Ng. 2016. Joint inference for event coreference resolution. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3264–3275.

Emily Manetta. 2018a. The structure of complex predicates in hindi-urdu: evidence from verb-phrase ellipsis. *Trends in Hindi Linguistics*, 325:47.

Emily Manetta. 2018b. Verb-phrase ellipsis and complex predicates in hindi-urdu. *Natural Language & Linguistic Theory*, pages 1–39.

Annie Montaut. 2016. Noun-verb complex predicates in hindi and the rise of non-canonical subjects. In *Approaches to Complex Predicates*, pages 142–174. Brill.

Amitabha Mukerjee, Ankit Soni, and Achla M Raina. 2006. Detecting complex predicates in hindi using pos projection across parallel corpora. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 28–35. Association for Computational Linguistics.

Martha Palmer, Rajesh Bhatt, Bhuvana Narasimhan, Owen Rambow, Dipti Misra Sharma, and Fei Xia. 2009. Hindi syntax: Annotating dependency, lexical predicate-argument structure, and phrase structure. In *The 7th International Conference on Natural Language Processing*, pages 14–17.

Julien Plu, Giuseppe Rizzo, and Raphaël Troncy. 2015. A hybrid approach for entity recognition and linking. In *Semantic Web Evaluation Challenge*, pages 28–39. Springer.

Julien Plu, Raphaël Troncy, and Giuseppe Rizzo. 2017. Adel@ oke 2017: a generic method for indexing knowledge bases for entity linking. In *Semantic Web Evaluation Challenge*, pages 49–55. Springer.

Nitin Ramrakhiyani and Prasenjit Majumder. 2013. Temporal expression recognition in hindi. In *Mining Intelligence and Knowledge Exploration*, pages 740–750. Springer.

Marco Rospocher, Marieke van Erp, Piek Vossen, Antske Fokkens, Itziar Aldabe, German Rigau, Aitor Soroa, Thomas Ploeger, and Tessel Bogaard. 2016. Building event-centric knowledge graphs from news. *Journal of Web Semantics*, 37:132–151.

Roser Saurí, Jessica Littman, Bob Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. 2006. Timeml annotation guidelines. *Version*, 1(1):31.

Manish Shrivastava and Pushpak Bhattacharyya. 2008. Hindi pos tagger using naive stemming: Harnessing morphological information without extensive linguistic knowledge. In *International Conference on NLP (ICON08), Pune, India*.

Ankush Soni, Sambhav Jain, and Dipti Misra Sharma. 2013. Exploring verb frames for sentence simplification in hindi. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1082–1086.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.

Juhi Tandon, Himani Chaudhry, Riyaz Ahmad Bhat, and Dipti Sharma. 2016. Conversion from paninian

karakas to universal dependencies for hindi dependency treebank. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 141–150.

Ashwini Vaidya, Sumeet Agarwal, and Martha Palmer. 2016. Linguistic features for hindi light verb construction identification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1320–1329.

Ashwini Vaidya, Owen Rambow, and Martha Palmer. 2019. Syntactic composition and selectional preferences in hindi light verb constructions. *LiLT (Linguistic Issues in Language Technology)*, 17.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1237–1247.

Ikuya Yamada, Hideaki Takeda, and Yoshiyasu Takefuji. 2015. Enhancing named entity recognition in twitter messages using entity linking. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 136–140.

Bishan Yang and Tom M Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 956–966.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base.

Weiguo Zheng, Lei Zou, Wei Peng, Xifeng Yan, Shaoxu Song, and Dongyan Zhao. 2016. Semantic sparql similarity search over rdf knowledge graphs. *Proceedings of the VLDB Endowment*, 9(11):840–851.

Language Modeling with NMT Query Translation for Amharic-Arabic Cross-Language Information Retrieval

Ibrahim Gashaw
Mangalore University
Mangalagangotri, Mangalore-574199
ibrahimug1@gmail.com

H L Shashirekha
Mangalore University
Mangalagangotri, Mangalore-574199
hlsrekha@gmail.com

Abstract

This paper describes our first experiment on Neural Machine Translation (NMT) based query translation for Amharic-Arabic Cross-Language Information Retrieval (CLIR) task to retrieve relevant documents from Amharic and Arabic text collections in response to a query expressed in the Amharic language. We used a pre-trained NMT model to map a query in the source language into an equivalent query in the target language. The relevant documents are then retrieved using a Language Modeling (LM) based retrieval algorithm. Experiments are conducted on four conventional IR models, namely Uni-gram and Bi-gram LM, Probabilistic model, and Vector Space Model (VSM). The results obtained illustrate that the proposed Uni-gram LM outperforms all other models for both Amharic and Arabic language document collections.

1 Introduction

Information Retrieval (IR) is the activity of retrieving relevant documents to information seekers from a collection of information resources such as text, images, videos, scanned documents, audio, and music as well. These resources can be structured, indexed, and navigated through Language Technology (LT), which includes computational methods that are specialized for analyzing, producing, modifying, and translating text and speech (Madankar et al., 2016). The increasing necessity for retrieval of multilingual documents in response to a query in any language opens up a new branch of IR called Cross-Language Information Retrieval (CLIR). Its goal is to accept the query in one language, transform it into a searchable format and provide an interface to allow a user to search and retrieve information in different languages as per their information need (Sourabh, 2013).

The Amharic language is the official language of Ethiopia spoken by 26.9% of Ethiopia's population as mother tongue and spoken by many people in Israel, Egypt, and Sweden. Arabic is a natural language spoken by 250 million people in 21 countries as the first language and serving as a second language in some Islamic countries. Ethiopia is one of the nations, which have more than 33.3% of the population who follow Islam, and they use the Arabic language to teach religion and for communication purposes. Arabic and Amharic languages belong to the Semitic family of languages, where the words in such languages are formed by modifying the root itself internally and not simply by the concatenation of affixes to word roots (Shashirekha and Gashaw, 2016).

Nowadays, it is widely used to solve CLIR problems for many language pairs. However, much of the research on this area has focused on European languages despite these languages being very rich in resources. So this study is aimed to develop the NMT query translation based Amharic-Arabic CLIR system.

An essential part of CLIR is mapping between query and document collections by translating queries to the target document language or the source document to the target document language. We follow the first approach to translate the query words by using a pre-trained NMT model. For the purpose of this translation, we have constructed a small parallel text corpus by modifying the existing monolingual Arabic and its equivalent translation of Amharic language text corpora available on Tanzile (Tiedemann, 2012),

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 56–64
Hyderabad, India, December 2019.

as Amharic-Arabic parallel text corpora are not available for MT task.

The rest of the paper is organized as follows. CLIR approaches are discussed in section 2. Related works are reviewed in Section 3. The proposed CLIR approach based on LM is described in Section 4. Resources and configurations of experiments for evaluating the system and the results are detailed in Section 5, followed by a conclusion in section 6.

2 CLIR Approaches

In CLIR, the query and the document collection needs to be mapped into a common representation to enable users to search and retrieve relevant documents across the language boundaries (Tune, 2015). Based on the resources used to map the query and the documents in different languages, CLIR approaches can be categorized as; Dictionary-based approach, Latent Semantic Indexing (LSI), Machine Translation (MT) approach, and Probabilistic-based approach (Raju et al., 2014).

2.1 Dictionary-based approaches

Dictionary-based approaches use either an automatically constructed bilingual Machine Readable Dictionaries (MRD), bilingual word lists, or other lexicon resources to translate the query terms to their target language equivalents. This approach offers a relatively cheap and easily applicable solution for large-scale document collection. Due to Out of Vocabulary (OOV), some words in a query may not be translated. Further, linguistic concepts such as polysemy and homonymy may introduce ambiguity in translation of words (Shashirekha and Gashaw, 2016)

2.2 LSI approach

In the LSI approach, the documents of the source language are represented in the language-independent LSI space. Similarly, a user query can be treated as a pseudo-document and represented as a vector in the same LSI space. Even though the performance of the LSI model is on par with the traditional vector space model, the cost of computing Singular Value Decomposition (SVD) of very large collections is high, and it makes a difference between different meanings of ambiguous terms according to their contexts of utilization (Nie, 2010).

2.3 Machine Translation approach

MT is a process of obtaining a target language text for a given source language text by using automatic techniques. MT can be used to translate the query, the document, or both into the same language, and the retrieval process could then be treated similar to a conventional IR system. However, MT systems require time and resources to develop and are still not widely or readily available for many language pairs (Madankar et al., 2016).

2.4 Probabilistic-based approaches

Probabilistic-based approaches include corpus-based methods which translate queries and language modeling which avoid translation of queries.

2.4.1 Corpus-based methods

Corpus-Based approaches use multilingual corpora which can be parallel corpora or comparable corpora. In this approach, queries are translated on the basis of multilingual terms extracted from parallel or comparable document collections. While parallel corpora contain translation-equivalent texts which contain direct translations of the same documents in different languages, comparable corpora contain texts of the same subject which are neither aligned nor direct translations of each other but composed in their respective languages independently (Tesfaye, 2010). It is available only in a few languages and more expensive to construct.

2.4.2 Language modeling approaches

A language model is a probability distribution over all possible sentences or other linguistic units in a language. While the classification of LM is not exhaustive, and a specific language model may belong to several types, LM can be categorized as uniform, finite state, grammar-based, n-gram, and Neural Language Model (NLM) (or continuous space LM) that might be feed-forward or recurrent (SWLG, 1997). Uniform LM uses the same probability for all words of the vocabulary of the sentences if the number of sentences is limited. In finite-

state LM, the set of legal word sequences is represented as a finite state network (or regular grammar) whose edges stand for the words that are assigned probabilities. Grammar-based LM is based on variants of stochastic context-free grammars or other phrase structure grammars.

Data scarcity is a significant problem in building language models, as most possible word sequences will not be observed in training. One solution to this problem is continuous representations, or embedding of words to make their predictions that help to alleviate the curse of dimensionality in LM. The main advantage of LM is to estimate the distribution of various natural language phenomena for language technologies such as speech, machine translation, document classification and routing, optical character recognition, information retrieval, handwriting recognition, spelling correction, etc. (Kim et al., 2016). Over-fitting (random error or noise instead of the underlying relationship when its test error is larger than its training error) is the main limitation in current LM for small size datasets (Jozefowicz et al., 2016).

3 Related works

Most of the researchers have studied CLIR works related to different language pairs. However, the only work reported on Amharic and Arabic languages pair is "Dictionary Based Amharic-Arabic Cross-Language Information Retrieval System" (Shashirekha and Gashaw, 2016). The performance was affected by incorrect translation due to out-of-dictionary words and unnormalized Arabic words; specifically, diacritics not mapped with the dictionary words, and the query was formulated by selecting words available in the dictionary.

Some of the prominent works reported on Amharic and Arabic languages paired with other languages are discussed below.

In bilingual Amharic-English Search Engine (Munye and Atnafu, 2012), limitation of word coverage includes a large-size commercial bilingual dictionary and on-line bilingual dictionary for query translation and short data size. The system can perform best only on the selected query terms which are available in the dictionary. The lack of electronic resources such as morphological analyzers and large MRD have forced A. Argaw (2005) to spend considerable time to develop those resources themselves.

Solving the problem of word sense disambiguation will enhance the effectiveness of CLIR systems. Andres Duque et al. (2015), studied to choose the best dictionary for Cross-Lingual Word Sense Disambiguation (CLWSD), which is focused only on English-Spanish cross-lingual disambiguation and the disambiguation task is dependent on the coverage of dictionary and corpus size. Query suggestion that exploits query logs and document collections by mapping the input query of French language to queries of English language in the query log of a search engine by W. Gao et al. (2007) showed the strong correspondence between the French input queries and English queries in the log, but languages may be more loosely correlated. For example, English and Amharic. M.Al-shuaili and M.Garvalho (2016), proposed a technique to map characters automatically from different languages into English, without human interference and prior knowledge of the language. While mapping helps transliterations of OOV names to have the same or, at least, very similar pronunciations in any language, word structure, and writing direction add complexity for character mapping and originality of the names also affects the result of character mapping.

In the Corpus-based CLIR system for Amharic and English language pairs (Tesfaye and Scannell, 2012), the size and the quality of document constructed highly affected the performance of the system. Nigussie Eyob (2013), have developed a corpus-based Afaan Oromo-Amharic CLIR system to enable Afaan Oromo speakers to retrieve Amharic information using Afaan Oromo queries. The scarcity of aligned corpus creates a problem of translation disambiguation, and the dictionary is limited to translate words only.

F. Türe et al. (2012), explores combination-of-evidence techniques for CLIR using three types of statistical translation models: context-independent token translation, token translation using phrase-dependent con-

texts, and token translation using sentence-dependent contexts. Experiments on retrieval of Arabic, Chinese, and French documents using English queries show that no one technique is optimal for all queries, but statistically significant improvements in Mean Average Precision (MAP) over strong baselines can be achieved by combining translation evidence from all three techniques.

In all the above-mentioned cases, the key element is the mechanism to map between languages that can be encoded in different forms as a data structure of the query and document-language term correspondences in an MRD or as an algorithm, such as an MT or machine transliteration system.

Nowadays, the direction of CLIR is on utilizing neural approaches. Quing Liu (2018), proposed a neural approach to English-Chinese CLIR, which consists of two parts; bilingual training data and Kernel-based Neural Ranking Model (K-NRM). External sources of translation knowledge are used to generate bilingual training data which is then fed into a kernel-based neural ranking model. The bilingual training approach outperforms traditional CLIR techniques given the same external translation knowledge sources. K-NRM learns translation relationships from bilingual training data by capturing soft-matches from bilingual term pairs and combine soft-matches to generate final scores with a set of bins. Kazuhiro Seki (2018) explores a neural network-based approach to compute similarities of English and Japanese language text. They focus on NMT models and examine the utility of an intermediate state. The intermediate state of input texts is indeed beneficial for computing cross-lingual similarity outperforming other approaches, including a strong machine translation baseline.

Many of CLIR works related to neural approaches are focused on neural ranking methods not directly using NMT for query translation. In this work, an NMT based query translation is employed to map between Amharic and Arabic Languages using traditional IR ranking methods.

4 Proposed Amharic-Arabic CLIR System

Traditional IR in cross-language environment settings mainly allows measuring the similarity between the information need (query) in source language and collection of documents in both languages. In a CLIR environment, queries and documents are written in two different languages. In order to match terms between the two languages, a retrieval system needs to establish a mapping between words in the query vocabulary and words in the document vocabulary.

Deep learning NMT is a recent approach of MT that produces high-quality translation results based on a massive amount of aligned parallel text corpora in both the source and target languages. Deep learning is part of a broader family of ML methods based on artificial neural networks (MemoQ, 2019). It allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have improved the state-of-the-art research in language translation (LeCun et al., 2015). NMT is one of the deep learning end-to-end learning approaches to MT that uses a large artificial neural network to predict the likelihood of a sequence of words, typically modeling entire sentences in a single integrated model. The advantage of this approach is that a single system can be trained directly on the source and target text no longer requiring the pipeline of specialized systems used in statistical MT. Many companies such as Google, Facebook, and Microsoft are already using NMT technology (Wu et al., 2016). NMT has recently shown promising results on multiple language pairs. Nowadays, it is widely used to solve translation problems in many languages. However, much of the research on this area has focused on European languages despite these languages being very rich in resources.

Our research has been focused on resolving query translation ambiguity. The open-source NMT system, called OpenNMT (Klein et al., 2017), which is an open-source toolkit for NMT, is used to construct the Amharic-Arabic NMT model. The pre-trained model is used to translate the text in Amharic to the

Arabic language. Once the query is translated into Arabic, standard IR algorithms can be used to retrieve the relevant documents from Amharic and Arabic document collections. As shown in Figure 1, prepossessing (tokenization, punctuation, and stop-word removal) is done for Amharic and Arabic document collections first. Then language models are produced for both languages, which will be used to estimate the query likelihood of the given query.

The search module is used to input Amharic language queries and retrieve relevant documents in both languages. A sample screenshot of the proposed system displaying relevant documents as a list of a hyperlink for a sample user query is shown in Figure 2.

A Sample Amharic text which is preprocesed after sentence spliting, tokenizing words, punctuation and stop-word removal is shown in Table 1, the same procedure is followed for Arabic text also.

A language model, which is a probability of words in each document $p(w|d)$ in the collection, is used to rank the documents according to the probability of generating the query. The query likelihood is given by $P(q|d) = \prod_{i=1}^{m} p(q_i|d)$. But this will assign zero probability for the words that are not available in the specific documents. Therefore the following maximization technique, which is LM with Jelineck-Mercer smoothing (Zhai and Lafferty, 2017), is used to optimize the likelihood of a given query, as shown in Equation 1.

$$prob(q_{t_i}) = \prod_{i=1}^{n} \lambda * p(q_{t_i}|m_d) + 1 - \lambda * p(q_{t_i}|m_c) \quad (1)$$

where, $prob(q_{t_i})$ is the probability of query term in position i, m_d is the probability in the document language model, m_c is the probability in the collection language model λ is the smoothing parameter and n is the length of the given query. After extensive experiments, λ is set to 0.9999. A document that is more likely to generate the user query is considered to be more relevant.

5 Experiments and Results

To design, develop, and maintain effective IR system, evaluation is very crucial as it allows the measurement of how successfully an information retrieval system meets its goal of helping users fulfill their information needs.

There are two approaches for evaluating the effectiveness of IR systems: (i) user-based evaluation and (ii) system-based evaluation. In the system-based evaluation method, several human experts evaluate the system to prepare a set of data that can be reused in later experiments. The user-based evaluation method quantifies the satisfaction of users by monitoring the user's interactions with the system (Samimi and Ravana, 2014). In this work, the focus is on system-oriented evaluation that focuses on measuring how well an IR system can rank the most relevant documents at the top for a given user query.

To evaluate the proposed Amharic-Arabic CLIR system, test collections (document corpus, search queries, and relevance judgments) have been prepared as bench-marked datasets are not available. Amharic is used as a source language to retrieve target language documents in Arabic as well as in Amharic. Experiments are conducted on four conventional IR models, namely Uni-gram and Bi-gram LM, Probabilistic model, and VSM. Unigram LM is the bag-of-words model where the probability of each word only depends on that word's own probability in the document. Bigram LM denotes n-gram models with n = 2. It is assumed that the probability of observing the i^{th} word w_i in the context history of the preceding $i-1^{th}$ word can be approximated by the probability of observing it in the preceding $n-1^{th}$ word. The Probabilistic model makes an estimation of the probability of finding if a document d_j is relevant to a query q, which assumes that the probability of relevance depends on the query and document representations. VSM is an algebraic model for representing queries and documents as vectors of identifiers.

Relevant judgments can be created using Crowdsourcing (Maddalena et al., 2016; Efron, 2009), (Ravana et al., 2015), which is a time-consuming and expensive task. Therefore, we considered the topmost ranked documents and took the union of all intersections between Unigram and Bigram, Unigram and VSM, Bigram and probability, and Probability and

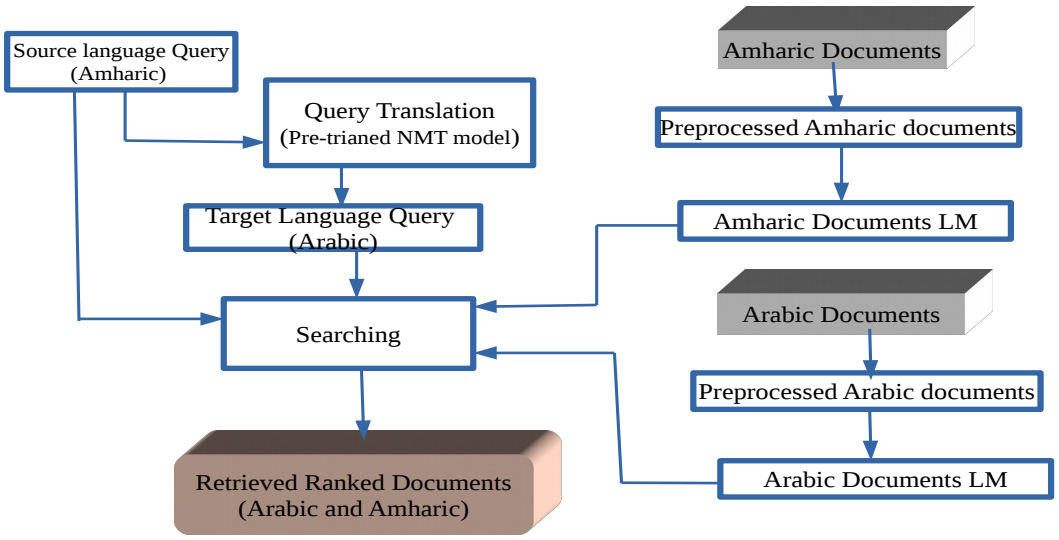


Figure 1: **Amahric-Arabic CLIR Architecture**

Table 1: Sample Amharic Text Preprocesing

Sample Amharic Text from Tanzile (Chapter 1)	**Preprocessed Text**
በአላህ ስም እጅግ በጣም ሩኅሩህ በጣም አዛኝ በኾነው፡፡ (1) ምስጋና ለአላህ ይገባው የዓለማት ጌታ ለኾነው፤ (2) እጅግ በጣም ርኅሩህ በጣም አዛኝ (3) የፍርዱ ቀን ባለቤት ለኾነው፡፡ (4) አንተን ብቻ እንግዛለን፤ አንተንም ብቻ እርዳታን እንለምናለን፡፡ (5) ቀጥተኛውን መንገድ ምራን፡፡ (6) የእነዚያን በነርሱ ላይ በነ የዋልክላቸውን በነሱ ላይ ያልተቆጣህባቸውንና ያልተሳሳቱትንም ሰዎች መንገድ (ምራን፤ በሉ)፡፡ (7)	በአላህ ሩኅሩህ አዛኝ ምስጋና ለአላህ የዓለማት ጌታ ርኅሩህ አዛኝ የፍርዱ እንግዛለን እርዳታን እንለምናለን ቀጥተኛውን መንገድ ምራን በነ የዋልክላቸውን ያልተቆጣህባቸውንና ያልተሳሳቱትንም ሰዎች መንገድ ምራን

VSM. If the number of documents in this set is less than 10, the symmetric difference of the uni-gram model is taken. As it is shown in Figure 3, the documents (Amtext1.txt, Amtext43.txt, Amtext27.txt, Amtext39.txt, Amtext26.txt, Amtext41.txt, Amtext81.txt, Amtext67.txt, Amtext28.txt, Amtext34.txt) are selected as the top-ranked documents relevant for the query **"ምስጋና ለአላህ ይገባው የዓለማት ጌታ ለኾነው"** (All praise is due to Allah, Lord of the worlds). For evaluation, we configure our test collection as 75 Amharic search queries, 114 Arabic and equivalent translation of Amharic documents (each verse of the Quran is organized as a single document), and relevant judgments are extracted using Equation 2. the description of this test collection is shown in Table 2. The test collection and parallel Amharic-Arabic text corpora used for translation will be provided on request.

$$X_{q_i} = (A \cap B) \cup (A \cap D) \cup (B \cap C) \cup (C \cap D) \quad (2)$$

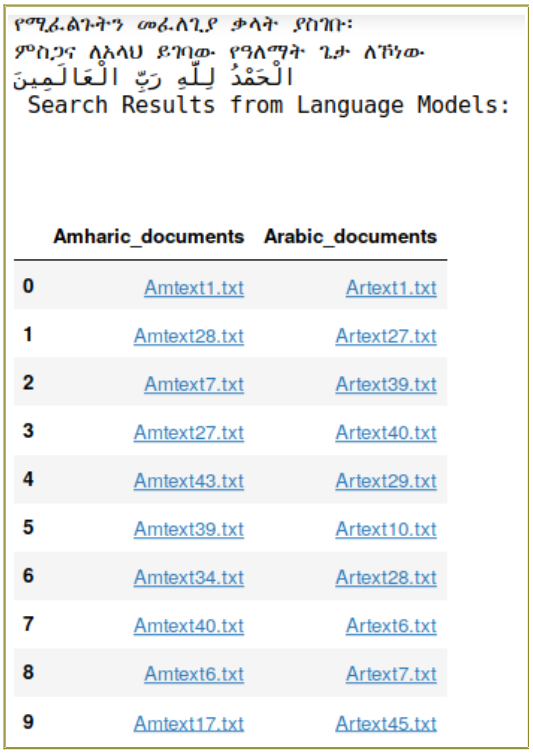
Table 2: Description of Test Collection for Amharic-Arabic CLIR Evaluation

#Query	#Documents (228)
75 Amharic Queries	114 separated Chapters of Quran in Arabic language
	114 separated Chapters of Quran in Amharic language

Top 10 documents judged as relevant for each query is computed as;

$$R_D = \begin{cases} X_{q_i}, & \text{if } X_{q_i} \geq 10 \\ X_{q_i} \cup A, & \text{if } X_{q_i} < 10 \end{cases} \quad (3)$$

where, R_D is list of relevant documents, X_{q_i} is set of top-ranked relevant documents for query i which is computed based on Equation 2, and A, B, C, and D are a set of top-ranked retrieved documents from Unigram, Bigram, Probability, and VSM models run. We adopted Text Retrieval Conference



	Amharic_documents	Arabic_documents
0	Amtext1.txt	Artext1.txt
1	Amtext28.txt	Artext27.txt
2	Amtext7.txt	Artext39.txt
3	Amtext27.txt	Artext40.txt
4	Amtext43.txt	Artext29.txt
5	Amtext39.txt	Artext10.txt
6	Amtext34.txt	Artext28.txt
7	Amtext40.txt	Artext6.txt
8	Amtext6.txt	Artext7.txt
9	Amtext17.txt	Artext45.txt

Figure 2: Display of relevant documents as a list of hyperlink for a sample users query

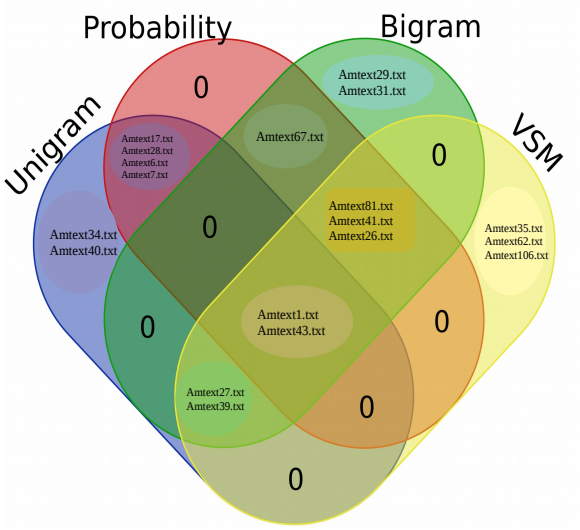


Figure 3: Four combinations of relevant judgment identification

(TREC) [1] test collection format where each query-document pair has a 5-level relevance scale, 0 to 4, with 4 meaning document d is most relevant to query Q and 0 meaning d is not relevant to Q.

The most frequently used and still the dominant approach to evaluating the performance of information retrieval systems are precision and recall. Precision is defined as the proportion of retrieved documents that are actually relevant, and recall is defined as the proportion of relevant documents that are actually retrieved. Both precision and recall can be expressed as; $Precision = \frac{\sum_{i=1}^{n} d_i}{n}$, and $Recall = \frac{\sum_{i=1}^{n} d_i}{R}$ where, d_i is the relevance level of the i^{th} document in the ranked output to a certain query, R is number of relevant documents for a query and n denotes the number of documents in the ranked output (Zhou and Yao, 2010).

Mean Average Precision (MAP) values are considered to give the best judgment in the presence of multiple queries. The evaluation metrics used in this work are; MAP and Recall.

MAP and recall are computed as the sum of Average Precision (AP) of each query divided by the number of queries and sum of average recall of each query divided by the number of queries, respectively.

The other measurement technique used for evaluation is Discount Cumulative Gain (DCG) that measures the usefulness or gain of a document based on its position in the result list. The gain is accumulated from the top of the result list to the bottom, with the gain of each result discounted at lower ranks. DCG adopted from Moffat and Zobel (2008) is accumulated at a particular rank position p as given in Equation 4.

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log_2(i+1)} \quad (4)$$

Comparing search algorithms performance from one query to the next cannot be consistently achieved using DCG alone. So the cumulative gain at each position for a chosen value of p should be normalized across queries. This is done by sorting all relevant documents in the corpus by their relative relevance, producing the maximum possible DCG through position p, also called Ideal DCG (IDCG) through that position (Chapelle and Wu, 2010) as shown in Equation 5.

$$nDCG_p = \frac{DCG_p}{IDCG_p} \quad (5)$$

[1] https://trec.nist.gov/

where,

$$IDCG_p = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

rel_i is the graded relevance of result at position i and $|REL|$ is the list of documents ordered by relevance in the corpus up to position p.

We also used Normalized NDCG to measure the usefulness of documents at first, fifth, and tenth position of ranked lists.

Evaluation Results of all models are presented in Table 3. In general, the proposed Unigram LM shows better performance than all others for both Amharic and Arabic language document collections. The unigram model makes a strong assumption that each word occurs independently, and consequently, the probability of a word sequence becomes the product of the probabilities of the individual words. Bigram model is better to identify the most relevant document at the top. As it is shown in Table 3, NDCG@1 has a higher value, which means it has a high cumulative gain in the first position. The bigram model considers the local context, which is the probability of a new word depending on the probability of the previous word. This Bigram model feature allows us to retrieve the most relevant document at the top. Still, it decreases the recall highly because it misses a strong assumption that each word occurs independently. Probability and VSM models perform almost the same. The length of the query influenced the final retrieval to a great extent both in Unigram and Bigram LM.

Table 3: Models Evaluation results

Models	NDCG@1	NDCG@5	NDCG@10	MAP	RECALL
Am-uni-gram	**0.9933**	**0.6969**	**0.7497**	**0.7866**	**0.8556**
Am-probability	0.97	0.6185	0.6279	0.5059	0.5867
Am-bi-gram	0.9733	0.4581	0.4426	0.2296	0.2681
Am-VSM	0.5233	0.5208	0.6038	0.5264	0.6504
Ar-uni-gram	**0.98**	**0.7896**	**0.8455**	**0.8202**	**0.8637**
Ar-probability	0.89	0.6812	0.6883	0.5698	0.64
Ar-bi-gram	0.9667	0.54	0.4827	0.2725	0.2844
Ar-VSM	0.3233	0.4483	0.5257	0.4148	0.5704

6 Conclusion

CLIR systems are very demanding and are directly connected with language-specific issues. The retrieval of relevant documents intended for further analysis is the first important step, which significantly influences the retrieval performance. We prepared Test collections (document corpus, search queries, and relevance judgments) as bench-marked data-sets are not available. Experiments are carried out on four conventional IR models, namely Unigram and Bigram LM, Probabilistic model, and VSM. The result illustrates that LM based CLIR performs better compared to others. Furthermore, we discovered that the length of the query influenced the final retrieval to a great extent. Our future directions towards achieving better results include experimenting on large data-sets with different domains because the document collection in this work is taken only from Quran, and explore recently introduced neural IR approaches Mitra et al. (2017).

References

Mazin Al-Shuaili and Marco Carvalho. 2016. Character Mapping for Cross-Language. International Journal of Future Computer and Communication, 5(1):18.

Atelach Alemu Argaw, Lars Asker, Rickard Coster, Jussi Karlgren, and Magnus Sahlgren. 2005. Dictionary-based amharic-french information retrieval. In Workshop of the Cross-Language Evaluation Forum for European Languages, pages 83–92. Springer.

Olivier Chapelle and Mingrui Wu. 2010. Gradient descent optimization of smoothed information retrieval metrics. Information retrieval, 13(3):216–235.

Andres Duque, Juan Martinez-Romo, and Lourdes Araujo. 2015. Choosing the best dictionary for Cross-Lingual Word Sense Disambiguation. Knowledge-Based Systems, 81:65–75.

Miles Efron. 2009. Using multiple query aspects to build test collections without human relevance judgments. In European Conference on Information Retrieval, pages 276–287. Springer.

Wei Gao, Cheng Niu, Jian-Yun Nie, Ming Zhou, Jian Hu, Kam-Fai Wong, and Hsiao-Wuen Hon. 2007. Cross-lingual query suggestion using query logs of different languages. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pages 463–470. ACM.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-Aware Neural Language Models. In AAAI, pages 2741–2749.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. arXiv preprint arXiv:1701.02810.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. nature, 521(7553):436.

Qing Liu. 2018. A Neural Approach to Cross-Lingual Information Retrieval. Ph.D. thesis, figshare.

Mangala Madankar, MB Chandak, and Nekita Chavhan. 2016. Information retrieval system and machine translation: a review. Procedia Computer Science, 78:845–850.

Eddy Maddalena, Marco Basaldella, Dario De Nart, Dante Degl'Innocenti, Stefano Mizzaro, and Gianluca Demartini. 2016. Crowdsourcing relevance assessments: The unexpected benefits of limiting the time to judge. In Fourth AAAI Conference on Human Computation and Crowdsourcing.

MemoQ. 2019. 5 Translation Technology Trends to Watch Out for in 2019.

Bhaskar Mitra and Nick Craswell. 2017. Neural models for information retrieval. arXiv preprint arXiv:1705.01509.

Alistair Moffat and Justin Zobel. 2008. Rank-biased precision for measurement of retrieval effectiveness. ACM Transactions on Information Systems (TOIS), 27(1):2.

Mequannint Munye and Solomon Atnafu. 2012. Amharic-English bilingual web search engine. In Proceedings of the International Conference on Management of Emergent Digital EcoSystems, pages 32–39. ACM.

Jian-Yun Nie. 2010. Cross-language information retrieval. Synthesis Lectures on Human Language Technologies, 3(1):1–125.

Eyob Nigussie. 2013. Afaan Oromo–Amharic Cross Lingual Information Retrieval. Ph.D. thesis, AAU.

BNV Narasimha Raju, MSVS Bhadri Raju, and KVV Satyanarayana. 2014. Translation approaches in cross language information retrieval. In International Conference on Computing and Communication Technologies, pages 1–4. IEEE.

Sri Devi Ravana, Prabha Rajagopal, and Vimala Balakrishnan. 2015. Ranking retrieval systems using pseudo relevance judgments. Aslib Journal of Information Management, 67(6):700–714.

Parnia Samimi and Sri Devi Ravana. 2014. Creation of reliable relevance judgments in information retrieval systems evaluation experimentation through crowdsourcing: a review. The Scientific World Journal, 2014.

Kazuhiro Seki. 2018. Exploring neural translation models for cross-lingual text similarity. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pages 1591–1594. ACM.

HL Shashirekha and Ibrahim Gashaw. 2016. Dictionary based amharic-arabic cross language information retrieval. In International Conference on Advances in Computer Science and Information Technology, pages 49–60.

Kumar Sourabh. 2013. An extensive literature review on clir and mt activities in india. International Journal of Scientific & Engineering Research.

EAGLES SWLG. 1997. Types of language models.

Aynalem Tesfaye and Kevin Scannell. 2012. Amharic–English Cross-lingual Information Retrieval: A Corpus Based Approach. Haramaya: Haramaya University.

Fasika Tesfaye. 2010. Phrasal Translation for Amharic English Cross Language Information Retrieval (Clir). Ph.D. thesis, AAU.

Jörg Tiedemann. 2012. Parallel Data, Tools and Interfaces in OPUS. In Lrec, volume 2012, pages 2214–2218.

Kula Kekeba Tune. 2015. Development of Cross-Language Information Retrieval for Resource-Scarce African Languages. Ph.D. thesis, International Institute of Information Technology, Hyderabad.

Ferhan Türe, Jimmy J Lin, and Douglas W Oard. 2012. Combining Statistical Translation Techniques for Cross-Language Information Retrieval. In COLING, pages 2685–2702.

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.

Chengxiang Zhai and John Lafferty. 2017. A study of smoothing methods for language models applied to ad hoc information retrieval. In ACM SIGIR Forum, volume 51, pages 268–276. ACM.

Bing Zhou and Yiyu Yao. 2010. Evaluating information retrieval system performance based on user preference. Journal of Intelligent Information Systems, 34(3):227–248.

Non-native Accent Partitioning for Speakers of Indian Regional Languages

G Radha Krishna
Dept. of E.C.E,VNRVJIET
Hyderabad, India
gunturrkr@gmail.com

R Krishnan
Amritha University
Coimbattore, India
drrkdrrk@gmail.com

V K Mittal
K L University
Vijayawada, India
drvinaykrmittal@gmail.com

Abstract

Acoustic features extracted from the speech signal can help in identifying speaker related multiple information such as *geographical origin*, *regional accent* and *nativity*. In this paper, classification of native speakers of South Indian languages is carried out based upon the accent of their non-native language, i.e., English. Four South Indian languages: *Kannada*, *Malayalam*, *Tamil*, and *Telugu* are examined. A database of English speech from the native speakers of these languages, along with the native language speech data was collected, from a non-overlapping set of speakers. Segment level acoustic features *Mel-frequency cepstral coefficients (MFCCs)* and F_0 are used. Accent partitioning of non-native English speech data is carried out using multiple classifiers: *k-nearest neighbour (KNN)*, *linear discriminant analysis (LDA)* and *support vector machine (SVM)*, for validation and comparison of results. Classification accuracies of 86.6% are observed using KNN, and 89.2% or more than 90% using SVM classifier. A study of acoustic feature F_0 contour, related to L_2 intonation, showed that native speakers of Kannada language are quite distinct as compared to those of *Tamil* or *Telugu* languages. It is also observed that identification of *Malayalam* and *Kannada* speakers from their English speech accent is relatively easier than *Telugu* or *Tamil* speakers.

1 Introduction

Identification of speakers, classification of their dialectal zones is important in a multilingual country like India (Bhattacharjee and Sarmah, 2012). Speaker uniqueness is manifested in both anatomical and learned traits. When the context is constrained, speaker characteristics can be used reliably to identify individuals (Arslan and Hansen, 1996). The *accent* is one of the glaring indications of linguistic and social background of a speaker. Studying the characteristics of dialect on a phonetic or phonemic level belongs to accent recognition (Mittal et al., 2014). Earlier studies have concluded that native language (L_1) affects the speaker's traits of their second language (L_2) (Ghorbani et al., 2018; Graham and Post, 2018). Analysis and classification of utterances that belong to specific groups of learners is the main objective of *Native Language Identification* (NLI) (Nisioi, 2015). However, there is very little research on the question of accuracy with which accent features can be used to identify a speaker's *regional* or *ethnic origin* (Harper and Maxwell, 2008). A solution to the problem of regional accent classification across English speaking South Indians is attempted in the present research, using a specifically developed corpus.

Discriminative classifiers based on characterizing acoustic differences across foreign accents can be employed to direct an accent dependent recognition system (Omar and Pelecanos, 2010; Ikeno and Hansen, 2006). Systems with an automatic evaluation of non-native speech, which includes characteristics of the mother tongue will have better performance over similar algorithms that depend upon target languages (Qian et al., 2017). This is particularly true when the text uttered is unknown. Native listeners are mostly aware of the speaker's regional accent and also the social or geographical subgroup within the region (Hanani et al., 2013). Automatic speaker characterization is vital in real-world applications and the advantages are widely open (Zampieri et al., 2017; Krishna and Krishnan, 2014).

Pattern recognition approach of collecting data, extracting suitable features, and training classification module using machine learning is a powerful tool in applications like Computer-Assisted-Pronunciation-Training (CAPT) programs. Acoustic descriptors are critical in tasks

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 65–74
Hyderabad, India, December 2019.

Table 1: Summary of data used for training and testing: (a) attributes (b) values for *training set* and (c) values for *testing set*

(a) *Attributes*	(b) *Training set*	(c) *Test set*
Total number of speakers	60	75
Speakers per language group (KAN, MAL*, TAM, TEL)	20	25
Speech Duration per speaker	300 sec	60 sec

Note: *MAL-Malayalam data set is used only in tests related to cepstral features.

such as sound Classification (Day and Nandi, 2007). State-of-the-art Accent Identification (AID) systems widely rely on spectral acoustic distribution for modeling the pronunciation. In applications like accent recognition, features distinguishing different phonemes of a language will be useful (Neumeyer et al., 2000). Language-specific differences in phonological development might be related to differences in phoneme and phoneme sequence frequency across languages (Ikeno and Hansen, 2006). Such variations are also represented by the intonation patterns of individuals (Mary and Yegnanarayana, 2008; Li et al., 2017). Apart from cepstral features that capture underlying acoustic characteristics, information from higher-level prosodic traits (Doddington, 2001; MALMASI and DRAS, 2017) were examined in the present study.

English is the most widely spoken second language in India and elsewhere in the world (Saha and Mandal, 2015; Guntur et al., 2018). Indian English has several varieties with their specific accents and phonological features and often a distinct lexicon. Research on spoken English of Indian speakers is urgently needed from a multidisciplinary perspective (Cheng et al., 2013; Krishna et al., 2019). Present work is aimed at comparing the acoustic properties that are likely to differ between English accents different groups of South Indian language of speakers. The non-native prosodic traits are a hindrance to proficiency in a second language (L_2), and also to the mutual understanding. Present work also examines the local prosodic changes in the non-native English speech, without incorporating any phonology of the specific languages. The ability to compensate against prosodic deviation during English production can be improved by identifying the articulatory gestures that emphasize the non-native speaker accent.

Table 2: Template of *file naming* for data recording

Native language	Name	Age / Sex	File Name

The paper is organized as follows: Section 2 presents the details of the database, including the recording methodology. Section 3 describes acoustic and prosodic features used in foreign accent recognition. Section 4 describes the classification procedures employed in the NLI experiments. Section 5 gives the details of the experiments and results. Analysis of results of regional accent classification is given in section 6. Section 7 describes the key outcome and contributions. Conclusions drawn are given in Section 8.

2 Data Sets of 4 Indian Regional Languages

The main focus of current research work is on differentiating the regional non-native English accents of speakers, and also describing foreign accent in terms of a common set of fundamental speech attributes. A database has been specifically developed (G.Radha Krishna and Mittal, 2018) with native and non-native speech samples containing utterance by the speakers belonging to language groups Kannada (KAN), Malayalam (MAL), Tamil (TAM), and Telugu (TEL). Table 2 shows the template of file naming process.

2.1 Selection of Regional Languages

Among more than six thousand languages in the world, less than 10% of the languages are spoken by more than 90% of the people. Speakers and learners of the English language constitute a large proportion in countries like India, South Africa, and much of the developing world. India has distinct linguistic communities, each of which shares a common language and culture. English, Hindi and dominant local languages are spoken non-natively by a large number of Indians. In South Indian cities, many people speak at least two second languages. It would be beneficial if speech based systems can store models of all known languages and carry out the task of NLI automatically.

Table 3: Summary of *speaker traits* and related *speech features* (Day and Nandi, 2007).

Speech characteristic	Speaker trait	Speech feature
Lexical, Syntactic (Idiolect, Semantics, Pronunciations, dictions, Idiosyncrasies)	Socio economic Educational status (Language use and sentence construction)	Vocabulary, Word arrangement & grammatical cues.
Prosodic (Rhythm, Intonation, Articulation rate etc.)	Personality type, Parental influences	Durational features. Pitch dynamics, Energy (likely to be Text / time dependent).
Low level acoustic features	Anatomical structure of speaker's vocal apparatus	Short-time spectrum, Predictor coefficients, Intensity, Pitch.

Table 4: Major *text-independent features* used in *prosodic analysis*.

Prosodic features	Factors that influences speech
Dynamics of F_0 contour	Identity of sound unit, its position from phrase, word; Speaking style; Intonation rules; Type of sentence (Interrogative, Declarative)
Intonation, Rhythm, Stress	Attitudinal, Accentual, Discourse, Grammatical

2.2 Speech Corpus Recording Methodology

The details of speech corpus developed for each of the languages is shown in Table 1. Native speech utterances of 20 speakers from each of the native language groups KAN, TAM, and TEL, each with a duration of 300 seconds formed the training set. English test samples for a duration of 60 seconds were collected from 25 speakers belonging to each of the four groups KAN, MAL, TAM, and TEL. As the sufficient number of native speakers of MAL are not readily available, it is included in the testing set only. The test utterances were recorded under identical conditions as training speech samples and there is no overlap between training and testing sets with respect to speakers and sentences. Each of the test samples is recorded for a duration of 60 seconds. The non-native English speech samples are collected from a set of speakers with nearly uniform geographical distribution within a region with an educational background of at least graduation, but who do not use English routinely.

Recordings of speakers were made in quiet office room conditions using Logitech h110 microphone and waveforms are sampled at a rate of 16 kHz. The recordings were made in a laboratory environment with written text, with negligible reverberation. The participants were asked to read aloud passages of a text from general topics. For applications like screening of non-native speech, read data can be used for both training and testing (Schuller et al., 2013). It is ensured that Gender weightages are equally distributed in training as well as testing data sets. The speakers in the training set are considered representative of the regional languages KAN, TAM and TEL. However, for testing set speakers of Malayalam were also included. These speakers are so chosen from language heartlands. The speakers in the test set are considered potential users of future systems augmented with automatic Accent Identification (AID) capability.

3 Features for Non-native Accent Partitioning

Understanding similar variations in foreign accents is a crucial factor for the development of an NLI system. The dominant articulatory traits of different languages are different (Koreman, 2018). In applications like accent recognition, features distinguishing different phonemes of a language will be useful (Li et al., 2013). The acoustic signature or the **voice individuality** of the speech signal are available as differences in transformations occurring at semantic, linguistic articulatory, and acoustic levels. Out of all the factors affecting speech, accent is a week factor in the sense that speech variation is not as evident as that due to speaker/gender.

Language-specific differences in phonological development might be related to differences in phoneme and phoneme sequence frequency across languages (Graham and Post, 2018). Speakers of the second language (SL) are expected to import certain patterns from their native language (NL)

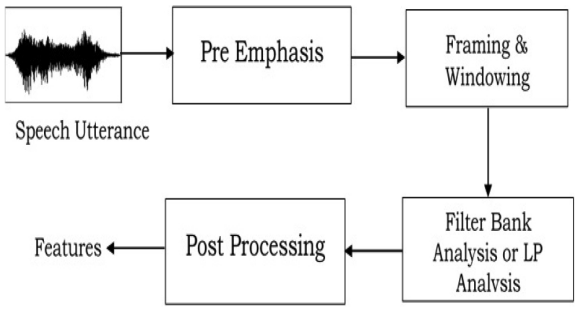


Figure 1: Front end *signal processing* for feature extraction

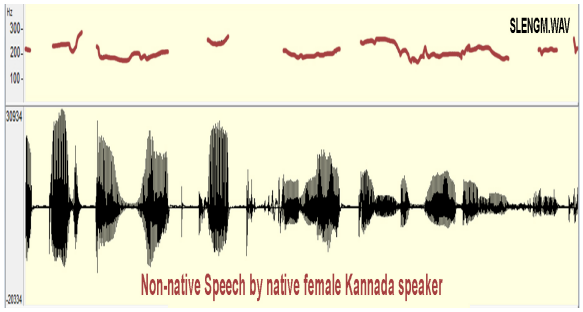


Figure 2: Waveform and Pitch contour of *non-native English speech* by female Kannada speaker

which are audible in SL. The influence of the surrounding speech prosody on new-born cry melody has been shown (Monnin and Loevenbruck, 2010). The non-native speech detection is thus very challenging .

Characterization of a foreign accent is mostly based on either auditory analysis or manual transcriptions of deviations. The auditory spectrum is consistent with several phenomena observed in speech perception and is useful in automatic speaker independent speech recognition. Features used for nonnativeness detection include cepstral vectors, phone strings and a variety of prosodic features, but when used alone, systems based on acoustic features perform better (Shriberg et al., 2005). We can consider acoustic features, which are proxy of phonetic reproduction as acoustic-phonetic features (Li et al., 2013).

3.1 Acoustic Features

Earlier investigations on text-independent non-native speech tied to underlying native language structure are based on (i) Global acoustic distribution of phonemes (which requires no language knowledge) (ii) Different intonations corresponding to uniqueness in the manner in which articulators are manipulated. The shape of the vocal tract is manifested in the envelope of the short-time power spectrum (Reynolds and Rose, 1995). The attributes that contain speaker identifiability for machine as well as for humans are of interest (Zheng et al., 2007; Franco et al., 2000).

In this study, acoustic features used for phonetic modeling of the accent differences consists of the cepstral features: Perceptive Linear Prediction Coefficients (PLPs), Linear Predictive Cepstral Coefficients (LPCCs), and MFCCs (Hermansky, 1990; Luengo et al., 2008; Mittal and Yegnanarayana, 2013). The steps followed are shown in Figure 1. Given all the alternative spectral features based on LPC - cepstrum and FFT cepstrum for speaker recognition, MFCCs, give a highly compact representation of the spectral envelope of a sound (López, 2014). The LPCCs are known to capture extra information from a speech that discriminates different languages. The PLPs which take advantage of psychoacoustic principles are robust against noise. A hierarchy of speech characteristics, related speaker traits, and possible speech features are listed in Table 3.

3.2 Prosodic Features

The prosodic structure is a critical aspect of language contact and gives important information related to the speaking habit of a person (Kinnunen and Li, 2010; Farrús et al., 2010). The goal is to capture prosodic idiosyncrasies of speakers belonging to different native languages. Prosodic cues Stress, Rhythm, and Intonation are each complex entities expressed using (i) Pitch (ii) Energy (iii) Duration. Major text-independent features used in prosodic analysis are given in Table 4.

In this study Prosodic statistics were obtained by performing different measurements of *pitch*, which are derived supra segmentally. The power of accent in voice identification is investigated as explained below. A *Generative model* of pronunciation describes what is acceptable, and *Discriminative model* both acceptable and unacceptable pronunciation, and the pronunciation score is the direct output of the classification module.

Non-native prosodic traits limit proficiency in a second language (L_2). Prosodic phenomena located on word level and above, help listeners to structure the speech signal and to process the linguistic content successfully. Table 4 shows some of the features useful for detecting non-native speech without annotation of prosodic events. The

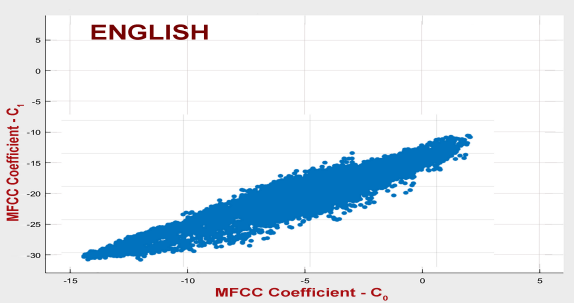


Figure 3: Distribution of MFCC Coefficients as a Scatter plot of C_0 versus C_1 for *native ENGLISH* speakers

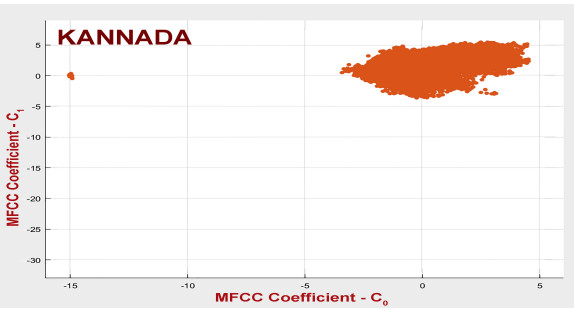


Figure 4: Distribution of MFCC Coefficients C_0 versus C_1 for *English speech by KANNADA speakers*

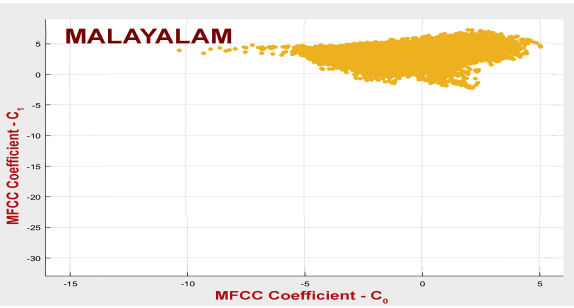


Figure 5: Distribution of MFCC Coefficients C_0 versus C_1 for *English speech by MALAYALAM speakers*

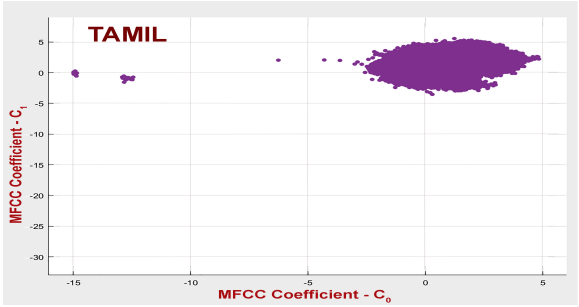


Figure 6: Distribution of MFCC Coefficients C_0 versus C_1 for *English speech by TAMIL speakers*

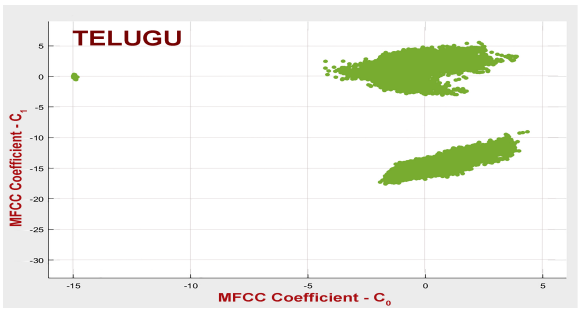


Figure 7: Distribution of MFCC Coefficients C_0 versus C_1 for *English speech by TELUGU speakers*

experiment by Rosenberg to foil a Speaker Verification system says that even an identical twin was unable to imitate the enrolled sibling well enough to get accepted by the system, tells the need to look at learned speaking behaviour.

4 Classification for Non-native Accent Partitioning

Speaker Classification can be conveniently defined as a grouping of speakers speaking in a similar manner, on the basis of acoustic characteristics (Chen et al., 2014). Classification of foreign accents directly from the acoustic features is attempted by using a test data set described in Table 1. The role of accent in voice identification is investigated as explained below. There exists a significant overlap between NLI approaches and computational methods for dialect and language identification (LID), and Support Vector Machine (SVM) classifiers are a very good fit for NLI (Zampieri et al., 2017).

4.1 Accent Partitioning using SVM Classifier

SVM is one of the most popular supervised classifiers on a wide range of data sets, which looks for a *maximum-margin hyper plane* for data separation (Wu et al., 2010; Bahari et al., 2013; Campbell et al., 2006). Accuracies of non-native accent classification were studied for the present problem by using the SVM classifier. The speech signal is first processed to extract attributes relevant to the *foreign accent* (Moustroufas and Digalakis, 2007). The most representative acoustic features, the LPCC, the PLP (Li et al., 2013) have been tested but were found to be less efficient. The input to the system is a 13 dimensional MFCC vector consisting of 12 cepstral coefficients and one energy coefficient. Thus the front end for the proposed classification system consisted of only 13 dimensional MFCC vector including C_0.

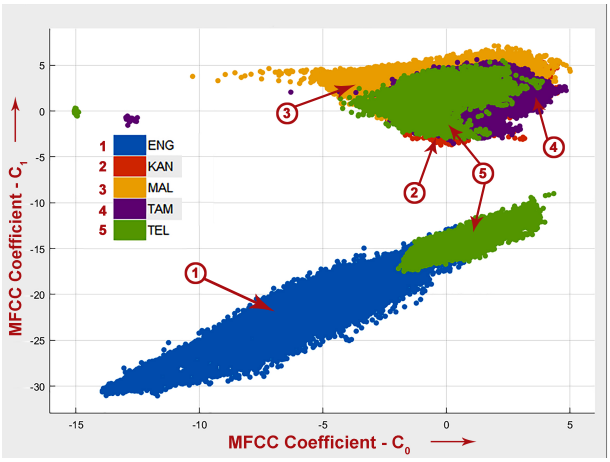


Figure 8: Distribution of MFCC Coefficients C_0 versus C_1 for *non-native English speech by four South Indian language speakers* against *native English speech.*

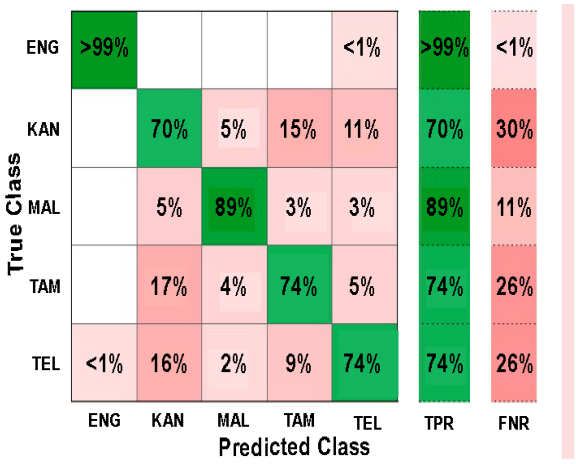


Figure 9: Confusion Matrix for SVM classification of *South-Indian English including native English*. **Note**: TPR is True Positive Rate, FNR is False Negative Rate.

4.2 Intonation Analysis

Native traits located at a word and sentence levels help listeners structure the speech signal. In many approaches that apply prosody to either *Language Identification* (LID) or *Speaker Recognition*, extracted features are based on statistics of pitch / energy contour segments or piecewise linear stylization of pitch / energy contours. Intonation is a key expressive factor which can covey the intent of a speaker, contains a lot more information than words and utterance (Ward et al., 2017). *Intonation* is more used than *energy* and *duration* features in the context of prosody. Listeners can discern a speaker's regional accent from intonation alone (Eady and Cooper, 1986; Tepperman and Narayanan, 2008).

Dynamics of F_0 contour corresponding to a sound is influenced by several factors such as the identity of the sound unit, its context, the speaking style of the speaker, intonation rules of the language, type of the sentence, etc. (Arias et al., 2010). The focus was mainly on the pitch since it is one of the most important characteristics of prosody and helps in predicting human intonation rating. These suprasegmental parameters can be used to model non-native English prosody (Hönig et al., 2012). In the present study, the main aim is to ascertain the influence of linguistic background on F_0 across regional varieties of English, future studies are planned to include the aperiodic components of excitation of expressive voices like Noh voice (Mittal and Yegnanarayana, 2015)

Table 5: *Non-native Regional English Accent Classification accuracies using (a) k-nearest neighbourhood (KNN), (b) Linear Discriminant (LDA), and (c) SVM*

Classifier	(a) *KNN*	(b) *LDA*	(c) *SVM*
Accuracy	**86.6%**	**82.5%**	**89.2**

5 Experiments and Results

To validate the hypothesis that the accent of the mother tongue is separable, experiments were performed to understand and to calibrate idiolectal differences in the non-native speech samples of the language groups KAN, MAL, TAM and TEL. The corpus is sampled at 16000 samples per second and the bit rate was 32 bits per sample. Silence removal has been implemented using a VAD algorithm (Kinnunen and Li, 2010). The feature vectors are computed over 20 msec windowed frames every 10 msec. Fourier spectra were computed for sequential frames 160 points apart by using a 320 point Hamming window. Finally Cepstral Mean Normalization (CMN) is applied by subtracting the mean value of each feature over the entire utterance. MFCCs are generated by windowing the signal, application of DFT, taking the log of the magnitude and warping the frequencies on Mel scale and finally application of DCT.

5.1 Non-native Accent Classification based upon Acoustic Features

Experiments were performed to establish the differences in the distribution of acoustic features in the non-native speech samples of four language groups KAN, MAL,TAM, and TEL. Graphical il-

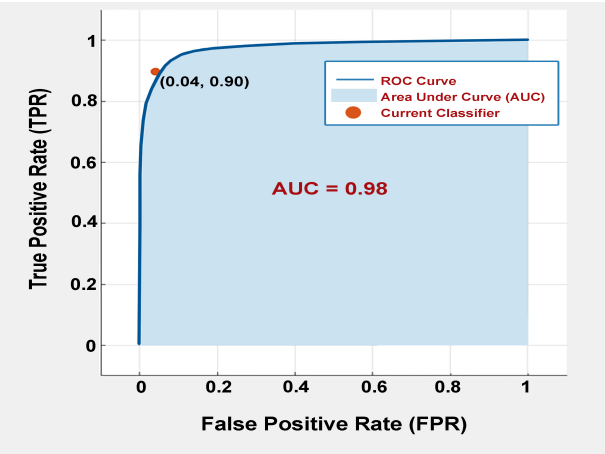


Figure 10: : ROC curve for SVM classification of *Non-native English speech by Kannada speakers*.

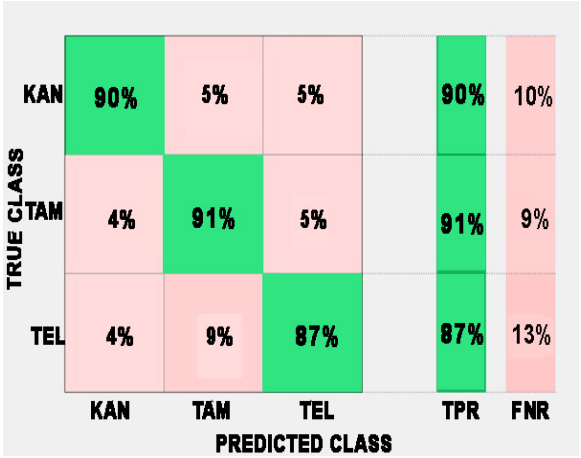


Figure 11: Confusion Matrix for SVM classification of *English by speakers of KAN, TAM, and TEL*. **Note**: TPR is True Positive Rate, FNR is False Negative Rate.

lustration of accent partitioning on test data is shown in Figures 3,4,5,6,7, and 8. It indicates that the high classification accuracies are possible in the present task. Classification of foreign accents directly from the acoustic features is attempted, by using data set described in Table 1. Figure 9 shows the confusion matrix for best performing SVM classifier for the five class classification. Figure 11 shows the confusion matrix for the three class classification.

The confusion matrix indicates that the identification rates for Kannada and Tamil language speakers from their non-native English speech can be high compared to that of Telugu native speakers . The Receiver Operating Point Curve (ROC) shown in Figure 10 is a plot of true positive rate as a function of false positive rate, which is very close to the upper left hand corner, indicates that the classifiers can achieve good overall accuracies.

Verification of accent partitioning of non-native speech using a series of classification techniques: k-nearest neighbourhood, and Linear Discriminant Analysis was also implemented. English speech samples of the native speakers of KAN, MAL, TAM, and TEL are tested against standard English speech corpus using TIMIT corpus. The resulting accuracies are 86.6% when a KNN clssifier is used, 82.5% when Discrimination classifier is used, and 89.2% using SVM classifier is used. These results are consolidated in Table 5. Figure 4, and 6 shows the corresponding confusion matrices, obtained during SVM classification.

5.2 Foreign Accent Discrimination based upon Prosodic Features

Experiments were conducted on native and non-native speech samples of bilingual and multilingual speakers. The pitch frequency was extracted using the ”pitch contour” function of the *Wave Surfer* software, and F_0 data was extracted. Typical waveform showing the non-native speech by a female Kannada speaker and the pitch contour were shown in Figure 2. The speakers in this study were asked to speak in their mother tongue or in English, and 20 exemplars were analysed from each group KAN, TAM, and TEL. In few cases the same speakers have spoken in other Indian language of the neighbouring state.

The difference in F_0 contour between native and non-native speech for speakers from each group has been tested. These results shown in Table 6 clearly indicate that the mean value of non-native pitch is markedly high in the case non-native speakers in all the three groups. The percentage deviation from native language to English speech for a group of 20 speakers in each of the three languages has been estimated and is presented in Table 7. It is evident from the scores presented in Table 7 that the dynamic variation of pitch is the least at 3.7% for the regional variant of KAN speakers, which is significantly less when compared to 9.5%,and 27% corresponding to native TAM and TEL speakers respectively.

Table 6: Mean (μ) and SD (σ) of *Pitch variation* of single speaker from three groups of native speakers when speaking (a) Native Language (NL) (b) English (c) Other South Indian language (OSIL)

L1	LANGUAGE SPOKEN					
	(a) NL		(b) English		(c) OSIL	
	μ	σ	μ	σ	μ	σ
Kan	214	32.2	254	32.3	235	32.4
Tam	227	21.7	248	28.9	230	30.6
Tel	133	21.5	157	22.9	150	26.3

6 Analysis of Results

- Figures 3,4,5, and 8 reveal that the English spoken by native Kannada and Malayalam speakers is distinct than native Tamil or Telugu speakers, when compared to standard English.
- Accent partitioning experiments from a short utterance of 60 seconds of test data, indicates the suitability of the SVM classifier, as can be seen from accuracies shown in Table 5.
- Figure 1 reveals that the English spoken by Telugu native speakers are marginally closer to standard English, compared to that of Kannada and Malayalam language speakers.
- Higher mean values of the non-native pitch shown in Table 6 indicates the accommodation of speakers of all native languages to suit different social groups.
- Table 7 shows that English speakers of Tamil and Telugu would produce statistically significant higher pitch contour deviations than KAN speakers.

7 Key Outcome and Contributions

- A framework to handle the deviations of L_2 influenced by closely related L_1s and to achieve better performance for a given NLI task, even with fewer features is proposed
- Current study is significant when the target languages are linguistically close, and large resources of spoken English are not available
- Prosodic differences across the South Indian English accents has been experimentally illustrated, which is useful in automatic intonation classification for L_2 speech acquisition.

Table 7: *Percentage increase in Standard Deviation* of pitch contour from native language speech to English speech (using two non-overlapping sets of 20 speakers from each native language group Kannada, Tamil, and Telugu).

Language group	Male	Female	Average
Kannada	0.9	6.5	3.7
Tamil	9	10	9.5
Telugu	33	21	27

- Present work helps in accurate recognition of regional accent, that can improve the speech and speaker recognition system performance.
- Distinct pitch pattern variations in non-native English speech by Malayalam, and Kannada speakers compared to that of Tamil and Telugu varieties can help in distinguishing them.

8 Conclusion

It can be concluded that the regional native language classification has been achieved with an accuracy of nearly 90%, by using the acoustic distribution of cepstral features on the four types of non-native South Indian English speech. It is known that systems make more mistakes among regionally close languages. Accent differences among the non-native speakers are reflected as the deviation of L_2 influenced by L_1 on prosodic level. Studies carried out based on intonation distribution indicates that English speaking South Indian groups corresponding to Kannada, Malayalam, Tamil, and Telugu are clearly divided as per their native languages. Prosodic differences in the native and English speech by South Indian speakers were detected without annotation. Present method can potentially be applied to other languages like Hindi, and in addressing the important question of finding a universal feature set for identifying the non-native speech.

Present research is useful in applications such as voice based wireless services like mobile health care, agriculture. Automatic accent characterization can also be applied to fields such as sociolinguistics and speech pathology. Future work can employ different speech styles, and characteristics of speaker population to be carefully scrutinized, and also by including multi-disciplinary information. Further, the results can be extended to separating language families and also for rating L_2 proficiency.

References

Juan Pablo Arias, Nestor Becerra Yoma, and Hiram Vivanco. 2010. Automatic intonation assessment for computer aided language learning. *Speech Communication*, 52(3):254–267.

Levent M. Arslan and John H.L. Hansen. 1996. Language accent classification in American English. *Speech Communication*, 18(4):353–367.

Mohamad Hasan Bahari, Rahim Saeidi, Hugo Van Hamme, and David Van Leeuwen. 2013. Accent Recognition Using I-vector , Gaussian Mean Supervector and Gaussian Posterior probability Supervector for Spontaneous Telephone Speech. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7344–7348.

Utpal Bhattacharjee and Kshirod Sarmah. 2012. Gmm-ubm based speaker verification in multilingual environments. *International Journal of Computer Science Issues (IJCSI)*, 9(6):373.

W. M. Campbell, D. E. Sturim, D. A. Reynolds, and A. Solomonoff. 2006. Svm based speaker verification using a gmm supervector kernel and nap variability compensation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I.

Nancy F. Chen, Sharon W. Tam, Wade Shen, and Joseph P. Campbell. 2014. Characterizing phonetic transformations and acoustic differences across English dialects. *IEEE Transactions on Audio, Speech and Language Processing*, 22(1):110–124.

Jian Cheng, Nikhil Bojja, and Xin Chen. 2013. Automatic accent quantification of indian speakers of english. In *INTERSPEECH*.

P. Day and A. K. Nandi. 2007. Robust text-independent speaker verification using genetic programming. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):285–295.

George R. Doddington. 2001. Speaker recognition based on idiolectal differences between speakers. In *INTERSPEECH*.

Stephen J. Eady and William E. Cooper. 1986. Speech intonation and focus location in matched statements and questions. *The Journal of the Acoustical Society of America*, 80(2):402–415.

Mireia Farrús, Michael Wagner, Daniel Erro, and Javier Hernando. 2010. Automatic speaker recognition as a measurement of voice imitation and conversion. *International Journal of Speech, Language and the Law*, 17(1):119–142.

Horacio Franco, Leonardo Neumeyer, Vassilios Digalakis, and Orith Ronen. 2000. Combination of machine scores for automatic grading of pronunciation quality. *Speech Communication*, 30(2):121–130.

Shahram Ghorbani, John H L Hansen, Robust Speech, and Systems Crss. 2018. Leveraging native language information for improved accented speech recognition. (September):2449–2453.

R. Krishnan G.Radha Krishna and Vinay Kumar Mittal. 2018. Native Language Identification from South Indian English Speech. In *Workshop on Machine Learning in Speech and Language Processing, September 7th, 2018*.

Calbert Graham and Brechtje Post. 2018. Second language acquisition of intonation: Peak alignment in American English. *Journal of Phonetics*, 66:1–14.

Radha Krishna Guntur, R Krishnan, and V.K. Mittal. 2018. Prosodic Analysis of Non-Native South Indian English Speech. In *Proc. The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages*, pages 71–75.

A. Hanani, M. J. Russell, and M. J. Carey. 2013. Human and computer recognition of regional accents and ethnic groups from British English speech. *Computer Speech and Language*, 27(1):59–74.

Mary P. Harper and Michael Maxwell. 2008. *Spoken Language Characterization*, pages 797–810. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hynek Hermansky. 1990. Perceptual linear predictive (plp) analysis of speech. *The Journal of the Acoustical Society of America*, 87(4):1738–1752.

Florian Hönig, Anton Batliner, and Elmar Nöth. 2012. Automatic Assessment of Non-Native Prosody Annotation, Modelling and Evaluation. *Proceedings of the International Symposium on Automatic Detection of Errors in Pronunciation Training (IS ADEPT)*, pages 21–30.

A. Ikeno and J.H.L. Hansen. 2006. Perceptual Recognition Cues in Native English Accent Variation: "Listener Accent, Perceived Accent, and Comprehension". *2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings*, 1:I–401–I–404.

Tomi Kinnunen and Haizhou Li. 2010. An overview of text-independent speaker recognition: From features to supervectors. *Speech Communication*, 52(1):12 – 40.

Jacques Koreman. 2018. Category similarity in multilingual pronunciation training. In *Proc. Interspeech 2018*, pages 2578–2582.

G. Radha Krishna and R. Krishnan. 2014. Influence of mother tongue on english accent. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 63–67, Goa, India. NLP Association of India.

G.Radha Krishna, R.Krishnan, and V.K.Mittal. 2019. An automated system for regional nativity identification of indian speakers from english speech. In

16th IEEE India Council International Conference INDICON 2019 (Accepted).

Haizhou Li, Bin Ma, and Kong Aik Lee. 2013. Spoken language recognition: From fundamentals to practice. *Proceedings of the IEEE*, 101(5):1136–1159.

Kun Li, Xixin Wu, and Helen Meng. 2017. Intonation classification for L2 English speech using multi-distribution deep neural networks. *Computer Speech and Language*, 43:18–33.

Jesús Antonio Villalba López. 2014. Advances on Speaker Recognition in non Collaborative Environments. page 311.

Iker Luengo, Eva Navas, Iñaki Sainz, Ibon Saratxaga, Jon Sanchez, Igor Odriozola, and Inma Hernaez. 2008. Text independent speaker identification in multilingual environments. In *LREC 2008*.

SHERVIN MALMASI and MARK DRAS. 2017. Multilingual native language identification. *Natural Language Engineering*, 23(2):163215.

Leena Mary and B. Yegnanarayana. 2008. Extraction and representation of prosodic features for language and speaker recognition. *Speech Communication*, 50(10):782–796.

V. K. Mittal and B. Yegnanarayana. 2013. Effect of glottal dynamics in the production of shouted speech. *The Journal of the Acoustical Society of America*, 133(5):3050–3061.

Vinay Kumar Mittal and B Yegnanarayana. 2015. Study of characteristics of aperiodicity in noh voices. *The Journal of the Acoustical Society of America*, 137(6):3411–3421.

Vinay Kumar Mittal, B Yegnanarayana, and Peri Bhaskararao. 2014. Study of the effects of vocal tract constriction on glottal vibration. *The Journal of the Acoustical Society of America*, 136(4):1932–1941.

Vinay Kumar Mittal and Bayya Yegnanarayana. 2014. Significance of aperiodicity in the pitch perception of expressive voices. In *INTERSPEECH*.

Julia Monnin and Hélène Loevenbruck. 2010. Language-specific influence on phoneme development: French and drehu data. In *INTERSPEECH*.

N. Moustroufas and V. Digalakis. 2007. Automatic pronunciation evaluation of foreign speakers using unknown text. *Computer Speech and Language*, 21(1):219–230.

Leonardo Neumeyer, Horacio Franco, Vassilios Digalakis, and Mitchel Weintraub. 2000. Automatic scoring of pronunciation quality. *Speech Communication*, 30(2):83–93.

Sergiu Nisioi. 2015. Feature analysis for native language identification. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9041:644–657.

Mohamed Kamal Omar and Jason Pelecanos. 2010. A novel approach to detecting non-native speakers and their native language. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 4398–4401.

Yao Qian, Keelan Evanini, Xinhao Wang, David Suendermann-Oeft, Robert A. Pugh, Patrick L. Lange, Hillary R. Molloy, and Frank K. Soong. 2017. Improving sub-phone modeling for better native language identification with non-native english speech. In *INTERSPEECH*.

D. A. Reynolds and R. C. Rose. 1995. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3(1):72–83.

Shambhu Nath Saha and Shyamal Kr. Das Mandal. 2015. Study of acoustic correlates of english lexical stress produced by native (l1) bengali speakers compared to native (l1) english speakers. In *INTERSPEECH*.

Björn Schuller, Stefan Steidl, Anton Batliner, Felix Burkhardt, Laurence Devillers, Christian Müller, and Shrikanth Narayanan. 2013. Paralinguistics in speech and language - State-of-the-art and the challenge. *Computer Speech and Language*, 27(1):4–39.

E. Shriberg, L. Ferrer, S. Kajarekar, A. Venkataraman, and A. Stolcke. 2005. Modeling prosodic feature sequences for speaker recognition. *Speech Communication*, 46(3-4):455–472.

Joseph Tepperman and Shrikanth Narayanan. 2008. Better nonnative intonation scores through prosodic theory. In *INTERSPEECH*.

Nigel G Ward, Nigelward@acm Org, Paola Gallardo, and Amanda Stent. 2017. Non-Native Differences in Prosodic-Construction Use. *Dialogue & Discourse*, 8(1):1–30.

Tingyao Wu, Jacques Duchateau, Jean Pierre Martens, and Dirk Van Compernolle. 2010. Feature subset selection for improved native accent identification. *Speech Communication*, 52(2):83–98.

Marcos Zampieri, Alina Maria Ciobanu, and Liviu P. Dinu. 2017. Native Language Identification on Text and Speech. 2013:398–404.

Nengheng Zheng, Tan Lee, and P. C. Ching. 2007. Integration of complementary acoustic features for speaker recognition. *IEEE Signal Processing Letters*, 14(3):181–184.

A Little Perturbation Makes a Difference: Treebank Augmentation by Perturbation Improves Transfer Parsing

Ayan Das and Sudeshna Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur,
Kharagpur, WB, India
ayan.das@cse.iitkgp.ernet.in
sudeshna@cse.iitkgp.ac.in

Abstract

We present an approach for cross-lingual transfer of dependency parser so that the parser trained on a single source language can more effectively cater to diverse target languages. In this work, we show that the cross-lingual performance of the parsers can be enhanced by over-generating the source language treebank. For this, the source language treebank is augmented with its perturbed version in which controlled perturbation is introduced in the parse trees by stochastically reordering the positions of the dependents with respect to their heads while keeping the structure of the parse trees unchanged. This enables the parser to capture diverse syntactic patterns in addition to those that are found in the source language. The resulting parser is found to more effectively parse target languages with different syntactic structures. With English as the source language, our system shows an average improvement of 6.7% and 7.7% in terms of UAS and LAS over 29 target languages compared to the baseline single source parser trained using unperturbed source language treebank. This also results in significant improvement over the transfer parser proposed by Ahmad et al. (2019) that involves an "order-free" parser algorithm.

1 Introduction

Cross-lingual dependency parsing involves training a dependency parser using a treebank in one language (source language) and applying it to parse sentences in another language (target language). This can be used to develop parsers for languages for which no treebank is available.

The syntactic similarity between the source and the target languages typically plays an important role in the success of a cross-lingual transfer parser (Zeman and Resnik, 2008; Naseem et al., 2012; Søgaard, 2011). A major challenge in transfer parsing is to bridge the difference in the syntax of the source and the target languages. For example, the object usually occurs after the corresponding verb in English while the verb normally occurs at the final position in a clause in Japanese.

In order to achieve better performance of the transfer parsers, researchers have worked on the selection of syntactically similar source languages for a given target language (Søgaard, 2011; Rasooli and Collins, 2017; Wang and Eisner, 2016). Attempts have also been made towards improving the performance of the transferred parsers for a given source-target language pair by reducing the syntactic gaps between them. This is done by transforming the source language parse trees (Aufrant et al., 2016; Rasooli and Collins, 2019; Wang and Eisner, 2016, 2018; Das and Sarkar, 2019) using knowledge of the typological properties of the target language. However, these approaches are target language specific and may not give satisfactory results for multiple languages.

Recent work by Ahmad et al. (2019) proposed an "order-free" parser model that comprises of a transformer-based encoder and a graph-based decoder. They show that the self-attention mechanism of the transformer with direction independent position encoding used in their model gives rise to improved performance for transfer between distant pair of languages compared to a standard parser model that uses an RNN based encoder and stack pointer-based decoder.

In this paper, we propose a different approach for enhancing the performance of a target language independent transfer parser based on a single source language by augmenting the treebank of the source language without using any target language information. For this, we add sentences obtained by rearranging the original sentences in the treebank while keeping the parse tree of the sentence fixed. This can be construed as generating

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 75–84
Hyderabad, India, December 2019.

a more general treebank which may contain sentences not conforming syntactically to the source language.

Specifically, we introduce controlled perturbation in the relative ordering of the head-dependent pairs in the source language parse trees. We stochastically alter the order of some of the head-dependent pairs in the source language sentences while keeping the head-dependent relations in the parse trees intact. This perturbation reduces the dependency of the parser on the word order in the training sentences and makes it more robust towards the variation in syntax.

We show that a stack-pointer network-based parser model (Ma et al., 2018) trained using this treebank results in improvement of the performance of the transfer parser over a baseline parsers trained on an unperturbed treebank. This parser also significantly outperforms the "order-free" parser model proposed by Ahmad et al. (2019) model by 3.8% UAS and 4.2% LAS. We also show that our target language independent approach gives a competitive performance with that of a target language specific transformation approach (Das and Sarkar, 2019).

2 Related Work

Initial work on model transfer involved training delexicalized models (Zeman and Resnik, 2008; McDonald et al., 2013) using only language independent non-lexical features such as PoS tags in the source language treebanks.

Several approaches for model transfer that incorporate lexical features in the transfer models have been reported in the literature. These include use of cross-lingual word clustering (Täckström et al., 2012), dictionary-based mapping of distributed word embeddings and projection-based bi-lingual word representations (Xiao and Guo, 2014; Guo et al., 2015; Schuster et al., 2019; Ahmad et al., 2019).

Søgaard (2011) proposed an approach for selecting training instances from source language by ranking them in terms of similarity with the target language sentences in terms of PoS tag perplexity. Naseem et al. (2012); Täckström et al. (2013); Zhang and Barzilay (2015) presented a multilingual algorithm for dependency parsing that selectively learns the aspects (some features listed in World Atlas of Language Structures (WALS) (Haspelmath, 2005)) of the source languages relevant to the target language and ties the model parameters accordingly.

Another approach for improving the performance of cross-lingual transfer parsers is by transforming the source language parse trees to match the syntax of the target language. Aufrant et al. (2016) improves performance of the transfer parsers by transforming the source language parse trees based on the knowledge of the target language syntax derived from WALS. Das and Sarkar (2019) also proposed a similar source language treebank transformation method in which knowledge of the syntax of a target language is derived from small number annotated target language parse trees. Wang and Eisner (2016) generated synthetic treebanks by altering the word order of the source language treebanks using knowledge of the distribution of the noun and verb dependents of other real-world languages from their respective treebanks. Wang and Eisner (2018) proposed an approach for learning an optimized permutation parameter using the given source language treebank and a gold PoS tag annotated corpus in the target language. This parameter set is then applied to permute the source language parse trees to approximately match the syntax of the target language. These methods are however target language specific and may not perform well for other languages.

Bhat et al. (2017) have shown that training a parser model using *scrambled* parse trees of sentences of one domain improves performance of the parser over a parser model trained using the original treebank on test sentences of another domain. They scrambled the parse trees of sentences from newswire data and tested on conversational data. The scrambled treebank consisted either of all possible permutations of a subset of the parse trees in the original treebank, or, a fixed number of permutations of all the parse trees, where the permuted parse trees with the lowest perplexity assigned by a language model are selected.

Ahmad et al. (2019) proposed a parser algorithm that improves the quality of transfer parser independent of the target language. They have compared the performance of combinations of different encoder-decoder architectures. They consider a bidirectional LSTM based encoder (order-sensitive) and a transformer-based encoder (order-free), and, two types of decoders, stack-pointer based (order-sensitive) and a biaffine graph-based

(order-free) and have shown that overall best cross-lingual performance of a parser across several target languages can be achieved using the combination of transformer-based encoder and graph-based decoder model. This system is expected to be agnostic to the word order of the source sentence and thus work effectively for a variety of target languages.

Multi-source transfer (McDonald et al., 2011; Rosa and Zabokrtsky, 2015) parsing approaches combine treebanks of multiple source languages to train cross-lingual transfer parsing models.

3 Perturbation of Source Language Parse Trees

3.1 Parse Tree Structure based Perturbation

We now discuss the details of our stochastic perturbation algorithm. We call this perturbation scheme as **PTSPert**. In order to introduce variation in word order in the source language parse trees, we apply perturbation on each parse tree in the treebank which randomly changes the relative ordering of some head-dependent word pairs in the sentence. For each node in the parse tree, we classify each of its dependents as either *pre-dependent* or *post-dependent* based on whether it appears before or after its head word in the sentence. During perturbation, we convert a pre-dependent to post-dependent and vice versa with some probability. The probability of altering the relative position of a dependent with respect to its head word in a sentence is referred to as *perturbation probability* (P).

The PTSPert algorithm takes the original source language parse tree T_s as input and returns the perturbed sentence as output.

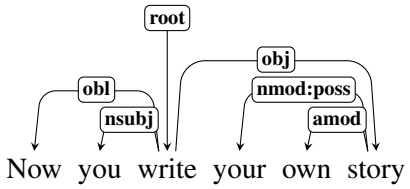
For each node n in the parse tree T_s, we maintain four lists: *pre-modifiers list* ($initpre_n$), *post-modifiers list* ($initpost_n$), *final pre-modifiers list* ($finalpre_n$) and *final post-modifiers list* ($finalpost_n$). The *pre-modifiers list* and *post-modifiers list* contain the pre-modifiers and post-modifiers of the node in the same sequence as they appear in the original sentence. The *final pre-modifiers list* and *final post-modifiers list* are initially *empty*.

The steps of the PTSPert algorithm are as follows;

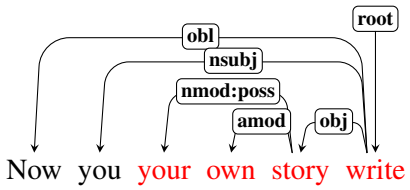
1. Traverse the words in the sentence from left to right. For each word in the sentence; let w be the node in T_s corresponding to the word.
   (a) Traverse $initpre_w$ from left to right. For each dependent in the list;
      i. With probability P, append the dependent to $finalpost_w$
      ii. With probability $1 - P$, append the dependent to $finalpre_w$.
   (b) Traverse $initpost_w$ from left to right. For each dependent in the list;
      i. With probability P, append the dependent to $finalpre_w$
      ii. With probability $1 - P$, append the dependent to $finalpost_w$.

The in-order traversal of the perturbed tree T_s based on the *finalpre*s and *finalpost*s of the nodes return the sentence with the new word-order.

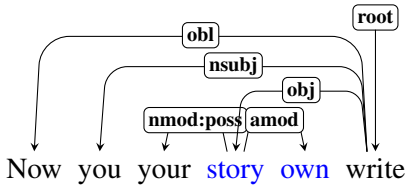
Example In Figure 2 we present the perturbed version of the sentence "Now you write your own story". The words whose positions have changed after perturbation are shown in red and blue. The final sentence after perturbation is "Now you your story own write".



(a) English sentence before perturbation



(b) Perturbation at "write"



(c) Perturbation at "story"

Figure 1: Perturbation on an English sentence.

After perturbation, the subtree with the word "story" as the head becomes a pre-dependent of the word "write". (Figure 1b) and the adjective "own" of "story" is converted to a post modifier. (Figure 1c)

3.2 Alternative Perturbation Models

Perturbation or introduction of noise in data is not new in natural language processing. It has been used to train a system to reconstruct the original sentence from its corrupted version (Dai and Le, 2015; Hill et al., 2016).

Artetxe et al. (2018) used a perturbation approach in unsupervised machine translation to learn the internal structure of a language and to reduce the dependence on the word order of the sentences to address the differences in the source and target languages. This was done by training an encoder-decoder system to recover the original sentence from its corrupted version given as input.

In this perturbation method, given a sentence of length N, N/k random swaps are made between the contiguous words, where k is a integer parameter. Artetxe et al. (2018) used $k = 2$. We call this perturbation approach **SwapPert**.

Some target language specific perturbation approaches extensively used in dependency parsing are discussed in Section 2.

4 Data and Parser Model

Data We carried out our experiments using treebanks of 29 languages from the UD v2.2 treebanks. We used the language-independent UD UPOS tags and dependency relations. We have used the acronyms of the language names in the rest of the paper. The full names of the languages are listed in Appendix A.1.

Word Embeddings We have used 300-dimensional fasttext (Bojanowski et al., 2017) pre-trained word embeddings for each language. The cross-lingual word embeddings were obtained by projecting the monolingual embeddings for all the languages into the space of the English language (Smith et al., 2017).

4.1 Parser

We have experimented with parser models with two types of encoder-decoder based parser models. The models are as follows;

- **RS:** Stack-pointer-based parser model (Ma et al., 2018) with BiLSTM RNN (Schuster and Paliwal, 1997; Hochreiter and Schmidhuber, 1997) based encoder and stack-pointer-based decoder model (Ma et al., 2018).
- **TG:** Transformer (Vaswani et al., 2017) based encoder with relative position representation (Shaw et al., 2018) and biaffine graph based decoder (Dozat and Manning, 2017). This encoder-decoder combination is due to Ahmad et al. (2019).

For our experiments, we have used the implementations of the parsers and the corresponding hyperparameter settings by Ahmad et al. (2019).[1]

5 Experiments and Results

We carried out the experiments corresponding to the different perturbation approaches under the following settings.

SwapPert Given a sentence of length N, for N/k perturbations, we have carried out separate experiments with $k = 2$ and $k = 10$. The stack-pointer-based parser model (Ma et al., 2018) (RS) was trained for this perturbation.

PTSPertRS This refers to the stack-pointer-based parser model (Ma et al., 2018) (RS) parser model trained using a source language treebank augmented with its versions perturbed by PTSPert. We experimented with different perturbation probability values ($P \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$).

STATtrans This refers to the stack-pointer-based parser model (Ma et al., 2018)(RS) parser model trained using source language treebank transformed using statistical knowledge of target language syntax derived from samples of 20 target language parse trees (Das and Sarkar, 2019). For each target language, we randomly sampled 20 parse trees from combined training and development sets. We trained separate models specific to each target language.

5.1 Baselines

RSUnpert This refers to the stack-pointer-based parser model (Ma et al., 2018)(RS) trained on unperturbed source language treebank.

TGUnpert This is the parser model comprising of a transformer-based encoder and a graph-based decoder (Ahmad et al., 2019) (TG) trained on unperturbed source language treebank.

All our experiments were repeated 5 times and we report the average result in this paper.

[1]The implementation was obtained from https://github.com/uclanlp/CrossLingualDepParser

TL	RSUnpert	TGUnpert	PTSPertRS (P=0.2)	STATtrans
en	**91.2/89.3**	90.3/88.4	91.2/89.3	89.9/87.6
no	**81.8/73.6**	80.3/72.2	81.1/73.5	79.7/72.4
sv	**82.3/74.3**	80.9/73.1	82.2/74.3	82.1/74.7
fr	76.1/70.7	78.6/73.4	**80.7/76.1**	79.9/75.0
pt	74.6/65.8	77.0/68.1	**79.1/69.8**	78.9/69.4
da	77.6/68.5	77.1/68.4	**78.1/69.3**	76.9/68.2
es	73.4/65.1	74.9/66.9	**77.4/69.1**	78.1/70.3
it	80.4/75.4	80.6/75.4	**83.9/79.4**	84.6/79.4
hr	61.1/51.5	62.4/52.5	**63.5/53.1**	66.7/56.8
ca	72.1/63.1	73.9/65.3	**76.0/66.9**	76.2/66.5
pl	72.5/60.1	75.4/62.8	**79.4/66.7**	80.9/69.2
uk	60.0/52.0	59.8/51.6	**62.6/53.3**	63.3/55.3
sl	68.0/56.4	68.6/56.6	**68.8/56.9**	69.5/57.4
bg	79.6/68.0	**80.1/68.7**	79.6/68.6	80.5/69.4
ru	61.2/52.2	61.4/51.9	**62.8/53.1**	64.4/55.3
de	69.2/59.3	72.0/62.1	**77.1/68.5**	78.6/69.7
he	56.4/45.0	55.9/46.9	**56.6/48.2**	58.2/50.6
cs	62.6/52.9	63.3/54.0	**64.8/54.3**	63.3/54.1
ro	61.9/50.6	66.3/55.1	**67.8/56.3**	69.9/59.3
sk	66.6/57.5	67.5/58.9	**69.4/59.2**	69.6/60.3
id	46.6/41.2	49.5/43.6	**55.0/47.8**	57.8/50.2
fi	**66.4/49.0**	66.6/48.6	66.0/48.6	66.5/49.2
et	64.6/44.1	**66.0/45.9**	64.1/44.9	67.1/47.6
zh*	41.3/24.2	40.3/24.0	**41.9/24.1**	44.8/28.4
ar	33.7/25.8	38.2/28.2	**43.3/33.6**	44.2/35.3
la	44.7/32.1	48.0/35.2	**51.3/37.1**	54.4/39.8
ko	33.6/14.4	**34.2/16.7**	33.9/16.4	39.3/21.5
hi	26.6/18.4	35.0/26.5	**45.0/35.9**	69.9/55.9
ja	15.0/9.3	27.2/19.4	**38.8/30.7**	60.8/46.8
Avg	62.1/52.0	63.8/53.8	**66.3/56.1**	68.8/58.5

Table 1: UAS%/LAS% corresponding to different perturbation methods on the target languages. '*' indicates the results corresponding to the delexicalized models. The underlined entries indicate the cases where *STATtrans* performs better than *PTSPertRS*

5.2 Results with English as Source Language

Evaluation Metric We report the results of our experiments in terms of unlabeled attachment score (UAS) and labelled attachment score (LAS) excluding punctuation and symbols.

In Table 1 we report the performance of the RSUnpert, TGUnpert and PTSPertRS ($P = 0.2$) and STATtrans on 29 target languages with English as the source language. The target languages are ordered according to their typological similarity with the English language based on the metric given by Ahmad et al. (2019). For the Chinese (zh) and Japanese (ja) languages, we report the results of the delexicalized transfer parsers for a fair comparison with the baseline. The best performance for PTSPertRS was achieved at P=0.2.

We observe that perturbation results in an overall improvement in the performance of the cross-lingual transfer parsers. Our proposed approach (*PTSPertRS*) performs better than the RSUnpert baseline parser in case of 24 out of 29 target languages. It improves cross-lingual performance of the transferred parser by **6.69%** and **7.74%** in terms UAS and LAS respectively. *PTSPertRS* also performs better than *TGUnpert* in case of 25 out of 29 target languages and improves average scores by 3.8%UAS and 4.2%LAS.

We also observe that although the *PTSPertRS* is a target language independent approach it gives better performance than *STATtrans* in case of 7 languages out of 29 target language. Furthermore, the parser model with transformer-based-encoder and graph-based-decoder (TG) trained using the treebank perturbed by PTSPert also performs better than *TGUnpert* and *RSUnpert*. However, it performs slightly worse than *PTSPertRS*.

In Table 2 we summarize the performance of the different approaches discussed in this paper in terms of UAS% and LAS% averaged over all 29 target languages with English as the source language. We observe that for different values of perturbation probability, PTSPertRS outperforms RSUnpert, TGUnpert and SwapPert. We also observe that SwapPert performs slightly better than RSUnpert for $k = 10$.

Consider the following German sentence (DE) and its English gloss (EN).

DE: *"Ich kann diese Tauch schule jeden empfehlen"*

EN: *I recommend this driving school to everyone.*

This is parsed by a transfer parser trained on English. The words and relations indicated in red show the errors by *RSUnpert* parser. The error is possibly because the verb *empfehlen* occurs at the end and after the object (Tauchschule), whereas the verbs occur before the objects in most English sentences. It is observed that the *PTSPert* parser correctly parses the sentence. This may have been made possible by perturbation of the source treebank resulting in instances of verb-final occurrences in the augmented treebank.

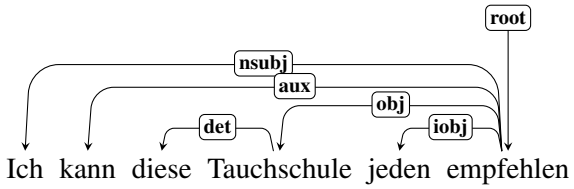
5.2.1 Dependency Relation-wise Analysis

In Table 3 we compare the labelled accuracies of PTSPertRS ($P = 0.2$) with RSUnpert and TGUnpert corresponding to 18 most frequent dependency relations averaged across all the 29 target languages. We observe that PTSPertRS performs better than RSUnpert and TGUnpert in terms of the *case, nmod, nsubj, amod, obl, advmod, acl, obj, aux, mark* and *cc* relations.

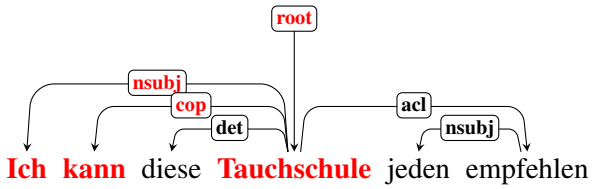
However, PTSPertRS performs worse than either RSUnpert or TGUnpert in terms of the *advcl,*

	RSUnpert	TGUnpert	PTSPert					SwapPert		STATtrans
			0.1	0.2	0.3	0.4	0.5	$N/2$	$N/10$	
UAS	62.1	63.8	65.8	**66.2**	66.1	66.1	64.8	59.7	63.6	68.7
LAS	52.0	53.8	55.7	**56.1**	55.6	55.6	54.4	49.3	53.7	58.4

Table 2: Comparison of average performance of different transfer approaches.

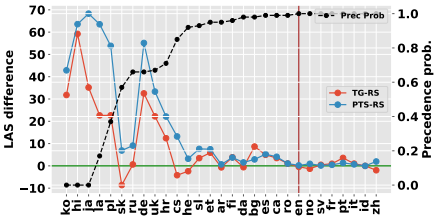


(a) Parser output of PTSPertRS model

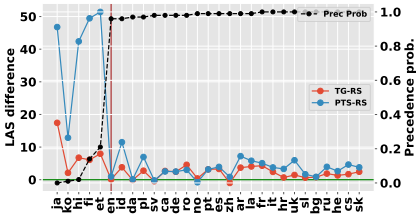


(b) Parser output of RSUnpert model

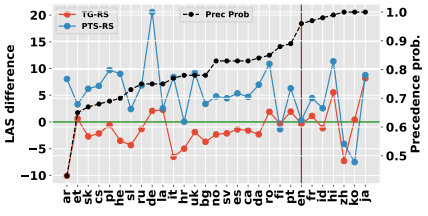
Figure 2: Parses of a German sentence.



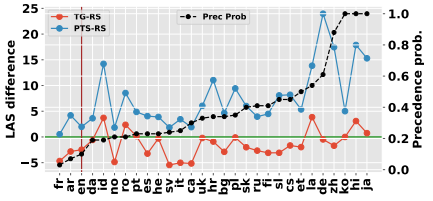
(a) Auxiliary dependency relation (*aux*)



(b) Case dependency relation (*case)*



(c) Subject dependency relation (*nsubj)*



(d) Oblique dependency relation (*obl)*

Figure 3: The blue and the red lines indicate the gain in LAS by PTSPertRS and TGUnpert over RSUnpert respectively. The black line indicates the precedence probabilities of the dependency relations in the language. The languages are sorted on the precedence probabilities from low to high. RS: RSUnpert, TG: TGUnpert, PTS: PTSPertRS

det, cop, nummod, compound, xcomp and *flat* relations. We note that the group of words related by *compound, fixed* and *flat* relations are usually arranged sequentially in a sentence and the dependents with *appos* relation always follow their respective heads. Thus perturbation with respect to these relations negatively affects the performance of the parsers. Furthermore, TGUnpert performs better than the PTSPertRS model in terms of the *det, nummod, cop, iobj* and *appos* relations. We observed that the dependents with *cop, nummod* and *det* relations appears before their head words in English. In case of the languages in which the copulas, determiners and numeric modifiers predominantly appears after their head words, the PTSPertRS shows an overall improvement of 17.25%, 4.14% and 50.0% respectively over the TGUnpert model. However, it loses out in terms of average accuracy in case of the other languages by 4.32%, 1.06% and 4.28% respectively. Since these relations appear before their respective heads in majority of the languages which includes English, the overall accuracy is less in terms of these relations.

For a dependency relation, we call probability of the dependents occurring before their heads in a language as the *precedence probability* of that relation in that language. The precedence probability of a relation in a language is measured by the ratio of the number of times the dependents with that relation appear before their heads and the total number of times the relation occurs in the data.

In our experiments, the precedence probabilities

Dep Rel	RSU-npert	TGU-npert	PTSPert-RS (P=0.2)
case	69.2	72.0	**75.6**
nmod	26.2	27.2	**29.0**
nsubj	52.5	51.2	**56.6**
amod	74.1	78.7	**79.4**
obl	40.8	38.8	**47.3**
advmod	63.2	60.7	**63.7**
obj	46.0	46.1	**48.8**
aux	59.5	72.4	**78.9**
mark	62.3	61.7	**63.5**
cc	71.3	71.1	**71.9**
acl	23.6	21.5	**24.8**
advcl	**32.5**	29.5	32.2
det	79.3	**86.2**	82.5
cop	57.4	**61.6**	60.1
nummod	65.3	**68.1**	67.7
compound	**36.5**	34.7	33.2
xcomp	34.9	**39.5**	34.9
flat	34.9	**35.6**	35.1

Table 3: Dependency-wise average accuracies of RSUnpert, TGUnpert and PTSPertRS (P=0.2).

of the relations in the source and target languages are estimated from the corresponding training and test sets respectively. Note that we have used these estimates for analysis of the results only.

In Figure 3 we compare the gain in LAS of PTSPertRS over RSUnpert parser corresponding to 4 different dependency relations over all the target languages. The dependency relations are chosen such that two are short distance relations (intra-phrase): *case* and *auxiliary* and two are relatively long-distance relations (inter-phrase): *nsubj* and *obl*.

For all the four dependency relations, we observe that the gains in performance of PTSPertRS over TGUnpert increases with the increase in the difference of precedence probability of the relations in the languages from that of English.

We also observe significant improvement in the performance of the PTSPertRS parsers over TGUnpert in case of the *nsubj* and *obl* for most of the language. Only in case of *fi* and *ko* languages, both RSUnpert and TGUnpert perform better than PTSPertRS in terms of the *nsubj* relation.

It is also observed that PTSPertRS performs significantly better than RSUnpert and TGUnpert in terms of the *aux* and *case* relations for the languages in which the precedence probabilities of the relations are different from that of English.

5.2.2 PTSPertRS with Variable Perturbation Probability Values

The results on PTSPertRS discussed above correspond to a single perturbation probability value applied on all the dependency relations. However, we observed that the best accuracies corresponding to different dependency relations were achieved at different P values. Thus we hypothesize that perturbing the dependents of different dependency relations by different amounts might be more helpful. We try to get an estimate of the perturbation probabilities corresponding different dependency relations from the performance of the PTSPertRS models trained using augmented treebanks perturbed with different perturbation probability values on the test set of a small number of languages. For this, we selected a random subset of 9 languages from the 29 languages. The 9 reference languages are *es, sl, he, id, sv, de, et, ar* and *hi*.

The steps for obtaining the probability value corresponding to a dependency relation are as follows;

- Corresponding to each P value in $\{0.0, 0.1, 0.2, 0.3, 0, 4, 0.5\}$, we find the average accuracy for the dependency relation over the 9 languages.
- We take the P value for the dependency relation for which the highest average accuracy is observed.

In Table 4 we present the perturbation probability values used for the different dependency relations. We apply these perturbation probability values corresponding to the different dependency relations to perturb the source language parse trees.

Dep Rels	Pert Prob
appos, parataxis, goeswith,flat, discourse, list, iobj,expl	0.0
det, cop, acl, advcl, mark, cc, compound	0.1
aux, ccomp, amod, obl, nummod, advmod, acl	0.2
nsubj, nmod, discourse, vocative	0.3
obj	0.4
case, csubj	0.5

Table 4: Perturbation probability values corresponding to the different dependency relations.

In Table 5 we present the performances of TGUnpert, RSUnpert, PTSPertRS with fixed P values and the PTSPertRS with variable P values averaged over all the 9 reference languages, 29 target languages and the 20 held-out languages respectively. On the set of the held-out 20 target

No. of languages		RSU-npert	TGU-npert	PTS-Pert (0.2)	PTS-Pert (Var. P)
9 reference languages	UAS	57.9	60.1	63.3	**64.7**
	LAS	47.7	50.0	53.2	**55.1**
20 held-out languages	UAS	64.0	65.5	67.6	**68.6**
	LAS	53.9	55.5	57.4	**58.6**
29 target languages	UAS	62.1	62.7	66.3	**67.4**
	LAS	52.0	52.5	56.1	**57.5**

Table 5: Average %UAS/%LAS over different sets of target languages for RSUnpert, TGUnpert, PTSPertRS (P=0.2) and PTSPertRS with variable P.

languages, we observe an improvement of 1.6% UAS and 2.16% LAS over the best single perturbation probability value of ($P = 0.2$) on the 20 languages. On the set of all the 29 languages also, this perturbation approach results in an overall improvement of 1.66% UAS and 2.49% LAS over the best single perturbation value ($P = 0.2$).

5.3 Results with Hindi as Source Language

We report here a summary of the results for Hindi as the source language. The variable perturbation probability values were derived from the following languages: *es, sl, he, id, sv, de, et, ar* and *en*.

In Table 6 we present the results corresponding to the different transfer approaches averaged over 29 target languages. We observe that PTSPertRS with different values of P outperform RSUnpert and TGUnpert. The best PTSPertRS result is achieved at P=0.3. PTSPertRS with variable P values also performs better than fixed P values. We observe that PTSPert with P=0.3 and variable P performs better than RSUnpert and TGUnpert for 27 out of 29 languages except *ko* and *ja*. We observe that *ko* and *ja* are syntactically quite close to Hindi and hence a parser model trained on unperturbed treebanks perform better than their perturbed versions.

In Table 7 we compare the performance of RSUnpert, TGUnpert, PTSPertRS with the $P = 0.3$ and PTSPertRS with variable P value averaged over all the 9 reference languages, 29 target languages and the 20 held-out languages respectively. We observe that PTSPertRS with variable P values gives the best results.

In Table 8 we report the average performance of RSUnpert, TGUnpert, PTSPertRS with the $P = 0.3$ and PTSPertRS with variable P values on Tamil (ta), Telugu (te), Urdu (ur) and Marathi (mr) languages for which treebanks are available in UD v2.2.

We observe that on an average over the four Indian languages, the best UAS and LAS scores are achieved for RSUnpert and TGUnpert respectively. Since the distribution of the dependents with respect to their heads for different dependency relations in the Indian languages are similar to that of Hindi, the best results are obtained for the parsers trained using unperturbed source treebank. This observation is in coherence with the results in English and Hindi where RSUnpert trained on unperturbed treebanks yield better results than PTSPert for the languages syntactically similar to the corresponding sources languages i.e. *no* and *sv* for English and *ko* and *ja* for Hindi.

5.4 Performance over Other Source Languages

We show that our perturbation approach enhances the performance of the "order-free" model proposed by Ahmad et al. (2019) for most of the 29 source languages. For this, we trained the parser model with transformer-based encoder and graph-based decoder using both unperturbed source language treebanks and PTSPert treebanks perturbed using P=0.1. The performance of the model trained using unperturbed treebank is taken as baseline. Following Ahmad et al. (2019), we trained the models using the first 4000 parse trees of each of the source language treebanks.

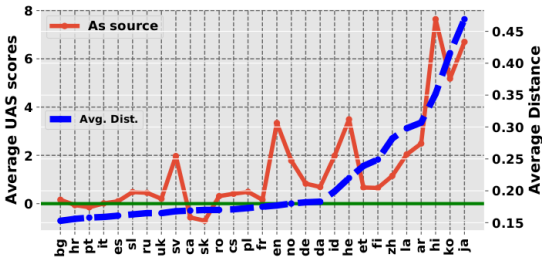


Figure 4: The red curve indicates average improvement over baseline for a language as source. The blue curve and the right y-axis indicates the average distance of a language from the rest.

In Figure 4, for each language as a source, we show the average improvement over all the target languages in cross-lingual performance of the parser trained using perturbed treebank. The languages are sorted according to their average syntactic distance from the other languages.

We observe that perturbation improves the average performance of the transfer parsers for the

	RSUnpert	TGUnpert	PTSPertRS					
			0.1	0.2	0.3	0.4	0.5	Var P
UAS	36.2	40.6	49.9	52.3	52.9	52.7	52.7	**54.0**
LAS	26.1	30.2	37.7	39.6	39.9	39.4	39.7	**41.2**

Table 6: Average UAS%/LAS% of different transfer parser approaches with Hindi as the source language.

No. of langs		RSU-npert	TGU-npert	PTS-Pert (0.3)	PTS-Pert (Var. P)
9 reference languages	UAS	31.8	36.7	50.8	**52.4**
	LAS	22.4	27.1	37.7	**39.6**
20 held-out languages	UAS	38.2	42.3	53.8	**54.7**
	LAS	27.8	31.6	40.8	**42.0**
29 target languages	UAS	36.2	40.5	52.9	**54.0**
	LAS	26.1	30.2	39.9	**41.2**

Table 7: Average %UAS/%LAS over different sets of target languages for different parsing approaches with Hindi as source language.

	RSU-npert	TGU-npert	PTS-Pert (0.3)	PTS-Pert (Var. P)
UAS	**75.9**	74.9	73.9	74.4
LAS	55.6	**55.9**	54.8	55.5

Table 8: Average %UAS/%LAS over *ta, te, mr* AND *ur* for different parsing approaches with Hindi as source language.

source languages except pt, sk and ca. The Pearson correlation coefficient of the average improvements with the languages as source with respect to the average distance from other languages is 0.82 indicating that the improvement due to perturbation is strongly correlated with the average distance from the target languages.

6 Conclusion

In this paper propose an approach for introducing perturbation in the source language treebank to improve single source target language independent cross-lingual transfer parsing. We show that this approach indeed helps to improve the performance of the transferred parsers over models trained using only source language treebanks.

References

Wasi Uddin Ahmad, Zhisong Zhang, Zuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2019. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. In *Proceedings of the 2019 Conference of the NAACL: HLT*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *International Conference on Learning Representations*.

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Zero-resource dependency parsing: Boosting delexicalized cross-lingual transfer with linguistic knowledge. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 119–130.

Riyaz A. Bhat, Irshad Bhat, and Dipti Sharma. 2017. Leveraging newswire treebanks for parsing conversational data with argument scrambling. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 61–66, Pisa, Italy. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

Ayan Das and Sudeshna Sarkar. 2019. Transform, combine, and transfer: Delexicalized transfer parser for low-resource languages. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 19(1):4:1–4:30.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th IJCNLP (Volume 1: Long Papers)*, pages 1234–1244, Beijing, China. Association for Computational Linguistics.

Martin Haspelmath. 2005. *The world atlas of language structures / edited by Martin Haspelmath ... [et al.]*. Oxford University Press Oxford.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 2: Short Papers)*, volume 2, pages 92–97.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the conference on EMNLP*, pages 62–72. Association for Computational Linguistics.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the ACL: Long Papers - Volume 1*, ACL '12, pages 629–637, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *Transactions of the Association for Computational Linguistics*, 5:279–293.

Mohammad Sadegh Rasooli and Michael Collins. 2019. Low-resource syntactic transfer with unsupervised source reordering. *CoRR*, abs/1903.05683.

Rudolf Rosa and Zdenek Zabokrtsky. 2015. Klcpos3-a language similarity measure for delexicalized parser transfer. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th IJCNLP (Volume 2: Short Papers)*, volume 2, pages 243–249.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. *arXiv preprint arXiv:1902.09492*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *CoRR*, abs/1803.02155.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the ACL: HLT: short papers-Volume 2*, pages 682–686. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. Proceedings of the 2013 Conference of the NAACL: HLT.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 conference of the NAACL: HLT*, pages 477–487. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Dingquan Wang and Jason Eisner. 2016. The galactic dependencies treebanks: Getting more data by synthesizing new languages. *Transactions of the Association for Computational Linguistics*, 4:491–505.

Dingquan Wang and Jason Eisner. 2018. Synthetic data made to order: The case of parsing. In *Proceedings of the 2018 Conference on EMNLP*, pages 1325–1337.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 119–129.

D. Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. *NLP for Less Privileged Languages*, pages 35 – 35.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. Association for Computational Linguistics.

A Appendices

A.1 Language Name Abbreviations

en - English, no - Norwegian, sv - Swedish, fr - French, pt - Portuguese, da - Danish, es - Spanish, it - Italian, hr - Croatian, ca - Catalan, pl - Polish, uk - Ukranian, sl - Slovenian, bg - Bulgarian, ru - Russian, de - German, he - Hebrew, cs - Czech, ro - Romanian, sk - Slovak, id - Indonesian, fi - Finnish, et - Estonian, zh - Chinese, ar - Arabic, la - Latin, ko - Korean, hi - Hindi, ja - Japanese.

Autism Speech Analysis Using Acoustic Features

Abhijit Mohanta
Indian Institute of Information Technology
Sri City, Chittoor, Andhra Pradesh, India
abhijit.mohanta@iiits.in

Vinay Kumar Mittal
Professor, K L University
Vijayawada, Andhra Pradesh, India
drvinaykrmittal@gmail.com

Abstract

Autism speech has distinct acoustic patterns, different from normal speech. Analyzing acoustic features derived from the speech of children affected with *autism spectrum disorder (ASD)* can help its early detection. In this study, a comparative analysis of the discriminating acoustic characteristics is carried out between *ASD affected* and *normal children* speech, from speech production point of view. Datasets of English speech of children affected with *ASD* and *normal* children were recorded. Changes in the speech production characteristics are examined using the *excitation source features* F0 and strength of excitation (SoE), the *vocal tract filter features* formants (F1 to F5) and dominant frequencies (FD1, FD2), and the *combined source-filter features* signal energy and zero-crossing rate. Changes in the acoustic features are compared in the five vowels regions of the English language. Significant changes in few acoustic features are observed for *ASD affected* speech as compared to *normal* speech. The differences between the *mean* values of the formants and dominant frequencies, for ASD affected and normal children, are highest for vowel /i/. It indicates that ASD affected children have possibly more difficulty in speaking the words with vowel /i/. This study can be helpful towards developing systems for automatic detection of ASD.

keywords: acoustic analyses of autism, autism spectrum disorder, ASD, dominant frequencies, formants

1 Introduction

ASD is a pervasive developmental disorder, defined clinically by observing the abnormalities in three areas: communication, social reciprocity, and hyperfocus or reduced behavioral flexibility (Kjelgaard and Tager-Flusberg, 2001; Diehl et al., 2009; McCann and Peppé, 2003). Study shows, at least 50% of the total population of ASD tends to show atypical acoustic patterns in their speech, and it persists throughout the improvement of other language aspects (DePape et al., 2012; Baltaxe and Simmons, 1985; Fusaroli et al., 2017). In fact, the exact characteristics of autism and its underlying mechanisms are also unclear (Kanner et al., 1943; Bonneh et al., 2011). According to study, 1 in 150 individuals with autism was reported in 2002, which became 1 in 68 in 2014 (Kumar et al., 2018; Autism and Investigators, 2014). It is reported that there are tens of millions of individuals with ASD worldwide, and it is affecting approximately 1.5% of our total population (Santos et al., 2013; Parish-Morris et al., 2016).

Communication impairments, abnormal voice quality and disturbances of prosody are some of the most important aspects among individuals with ASD who speak (Paul et al., 2005; Bonneh et al., 2011). Individuals with ASD speak with distinctive acoustic patterns in their speech, and as a result they face social interaction deficits (Fusaroli et al., 2017). The reason behind the language impairment in autism is the result of primary linguistic disorder with a focus on pragmatic impairments (Baltaxe, 1977). Besides, the speech signal of the children with ASD is reported as improperly modulated, wooden, and dull (Baltaxe and Simmons, 1985). In fact, in many cases, a significant spoken language delay and repetitive language can also be encountered (Mower et al., 2011). In general, normal children start establishing their vocabularies at the age of two years, whereas the children with ASD may not be able to do the same (Tager-Flusberg et al., 2005; Short and Schopler, 1988).

Previous studies mostly based on either speech prosody or unusual suprasegmental features of speech production of children with ASD (Bonneh et al., 2011). Like, in Shriberg et al. (2001), authors had reported the segmental and suprasegmental speech features of individuals with high-

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 85–94
Hyderabad, India, December 2019.

functioning autism (HFA). Also, some studies used objective measures to quantify speech related issues in autism (Bonneh et al., 2011). Some of the most significant analyses based on pitch features of individuals with ASD were reported in Brisson et al. (2014), Quigley et al. (2016), etc., where in each study authors had reported different result from others. For instance, in Brisson et al. (2014), authors had reported higher pitch value for ASD children as compared with normal children. On the other hand, in Quigley et al. (2016), authors had reported lower pitch value for ASD children as compared with normal children. Besides, in the case of the intensity based analyses, some of the studies indicated no significant differences between ASD and normal children (Quigley et al., 2016; Hubbard and Trauner, 2007). Likewise, based on duration (syllable duration, utterance duration, etc.), voice patterns, speech rate, etc., researchers had done some significant analyses on individuals with ASD (Santos et al., 2013; Kakihara et al., 2015; Bone et al., 2013). But, none of the previous studies had done only on English vowels, especially pronounced by non-native Indian English speakers with ASD. Also, many robust speech features like dominant frequencies (FD1, FD2), strength of excitation (SoE), etc., had not been considered in previous studies. Therefore, in this study, we have considered all these mentioned points.

This paper analyzed the *autism speech*, i.e., the speech signal of the children with ASD, by differentiating them from the normal children. Differences are made in terms of the speech production features of the ASD and the normal children. Here, only English vowels, i.e., /a/, /e/, /i/, /o/, and /u/ are taken into consideration, because of their relatively longer duration in the case of children with ASD. Also, the production of vowels sounds by an individual is not a random process; hence it is important to find characteristics of the speech production mechanism of children with ASD during the pronunciation of vowels sounds. This study on analyzing the speech production characteristics of the children with ASD has high importance, because it may play a vital role in improving the communication impairments associated with ASD. In addition, current diagnostic criteria for ASD do not include any atypical vocalizations (Bonneh et al., 2011). Hence, this study can be utilized as a diagnostic marker to identify ASD.

Table 1: Dataset Details of the *ASD* and the *Normal* Children

Attributes	*Group*	*Statistics*	
		Male	*Female*
Total children	ASD	11	02
	Normal	11	09
Age (in years)	ASD	03 to 09	3.5
	Normal	03 to 09	3.5 to 09
Reading skill (English)	ASD	Beginner	Beginner
	Normal	Beginner	Beginner
Data Duration (in sec)	ASD	6850	2500
	Normal	6000	6000

This study consists of four major steps. Firstly, two speech signal datasets were collected, by recording the sound files of the ASD and the normal children. Secondly, unwanted signal parts were removed, and the speech signal files were arranged in two different databases for the ASD and the normal children. Thirdly, speech signal processing methods were applied on the collected datasets to extract the selected production features. Finally, results were made by differentiating between the ASD and the normal children in terms of their speech production features.

The rest of the paper is organized as follows. Details about the two collected datasets of the ASD and the normal children are discussed in Section 2. Next, the signal processing methods and features used for analyses are discussed in Section 3. Section 4 presents key results and observations on results. Then, Section 5 discusses the analyses of observed results in speech production point of view. Section 6 represents key contributions. Lastly, Section 7 presents conclusions, along with the scope of future work on this topic.

2 Speech Datasets of ASD and Normal Children

Two speech signal datasets in the English language were recorded for this study, where one dataset contains the speech samples of 13 children with ASD, and another dataset contains the speech samples of 20 normal children. Details of both the datasets are given in Table 1. In this study, the number of ASD and normal children is different. There are numerous previous studies like Parish-Morris et al. (2016), Nakai et al. (2014), etc., where researchers took a different number of ASD

and normal children. Besides, children with age less than 3 years were not considered in this study, because typically the diagnosis of ASD starts by the age of 3 years when a child begins to show delays in developmental milestones (Santos et al., 2013; McCann and Peppé, 2003). Another reason was that the current study only focused on verbal children. Besides, in the case of the children with ASD, it was made sure by a well-experienced doctor and a psychologist that the children considered were diagnosed with ASD. The children with ASD considered for the data collection met the DSM-IV diagnostic criteria (Wing et al., 2011; Lord et al., 1994). Furthermore, all the children with ASD considered here had distinctive acoustic patterns in their speech, during the entire period of data collection. However, the normal children did not have any such issues and were living a normal life.

Speech samples were recorded every week (once or twice), for a period of over 1 year. Recordings took place in a noise-free empty room, which did not have any object that could distract the children. Also, the neutral emotional state of the children was affirmed during all the data collection sessions. The ASD and the normal children were asked to name in English a set of 25 specifically selected daily life pictures, shown to them along with each picture's name in English on a laptop. The pictures consisted of animals, vegetables, flowers, and English numbers. All the children were asked to pronounce only the object's name as a word, presented to them in the form of a picture. The children's first response was confronted by asking them to pronounce the picture's name. Then, we kept changing the pictures one by one, while the children named the object shown as a picture. Each child was asked to name the same set of pictures over each of the recording sessions. Five different pictures were selected for each of five English vowels, and the names of all the pictures were either in consonant-vowel-consonant (CVC) or consonant-vowel-vowel-consonant (CVVC) word format. The total utterances of 25 words by each child (5 vowels $\times$ 5 words) were recorded in each of the two such sessions, in a day.

Roland R-26 digital audio recorder was used with 48 KHz sampling rate to record the speech samples. The distance of 25 cm was maintained between the recorder and the speaker's mouth.

Our collected datasets have immense importance because of several reasons. Firstly, all the children considered here were non-native Indian English speakers. Whereas, in previous studies like Oller et al. (2010), Asgari et al. (2013), Marchi et al. (2015), Kakihara et al. (2015), etc., authors had not considered non-native Indian English speakers(children) with ASD. Secondly, in previous studies datasets were mostly collected from social interaction (Santos et al., 2013), constrained production (Bone et al., 2013) and spontaneous production (Fusaroli et al., 2017). But, here the datasets were recorded differently, as described earlier in this section.

3 Signal Processing Methods and Features

The production characteristics of speech signal of the ASD and the normal children are differentiated by examining changes in the source features, vocal tract system features and combined source-filter features. The source features F0 and strength of excitation (SoE), and the vocal tract filter features dominant frequencies (FD1, FD2) and first five formants (F1 to F5) are examined. The combined source-filter features signal energy (E) and zero-crossing rate (ZCR) are also examined. Here, for each speech feature, the mean (μ) or average values are computed. The mean values are computed for each English vowel by taking the average of all the calculated values of a particular speech feature, and this procedure is followed for each speaker. Besides, the μ_{SoE}, μ_E and μ_{ZCR} values are multiplied by 100, 1000, and 1000, respectively, for a better understanding.

3.1 Excitation Source Features

The excitation source feature F0 was derived using zero-frequency filtering (ZFF) method (Murty and Yegnanarayana, 2008; Yegnanarayana and Murty, 2009). The ZFF method involves computing the output of the cascade of two zero-frequency resonators (ZFRs). That is $y_1[n] = -\sum_{k=1}^{2} a_k y_1[n-k] + x[n]$ and $y_2[n] = -\sum_{k=1}^{2} a_k y_2[n-k] + y_1[n]$. Where, x[n] is pre-processed input signal, $a_1 = -2$ and $a_2 = 1$. This operation is repeated twice (denoted as $y_1[n]$ and $y_2[n]$) for a cascade of ZFRs. The trend in this output is removed by subtracting the moving average corresponding to the 10 ms window at each sample. The resultant trend removed signal, called the ZFF signal, given as $y[n] = y_2[n] -$

Table 2: Mean (μ) Values of the Source Features (*F0* and *SoE*), Combined Source-filter Features (*Energy E* and *Zero-crossing Rate ZCR*) and Vocal Tract Filter Features (*Formants Frequencies* and *Dominant Frequencies*) of the *Male* Children with *ASD* and *Normal* (Nm): (a) Acoustic Features and (b)-(f) Mean Values for Five English Vowels; *F1* to *F5* Indicate First Five *Formants Frequencies*, Respectively, and *FD1* and *FD2* are First and Second *Dominant Frequencies*, Respectively

(a) Features	*(b) /a/*		*(c) /e/*		*(d) /i/*		*(e) /o/*		*(f) /u/*	
	ASD	*Nm*	*ASD*	*Nm*	*ASD*	*Nm*	*ASD*	*Nm*	*ASD*	*Nm*
F0 (Hz)	263	258	267	260	271	262	269	256	246	236
SoE×100	34.9	32.1	43.4	46.3	44.0	47.7	39.1	35.7	35.9	31.1
E×1000	36.5	24.7	31.4	30.7	43.2	33.7	41.2	35.6	54.3	34.9
ZCR×1000	37.7	39.3	28.2	34.7	30.9	30.9	28.2	32.0	30.5	33.6
F1 (Hz)	720	453	554	452	589	424	657	557	662	498
F2 (Hz)	1628	1238	1665	1207	1658	1255	1310	1111	1466	1185
F3 (Hz)	2694	2486	2726	2551	2686	2566	2603	2504	2673	2446
F4 (Hz)	3712	3552	3715	3613	3675	3642	3561	3572	3603	3651
F5 (Hz)	4471	4455	4467	4435	4427	4425	4394	4320	4410	4331
FD1 (Hz)	1042	819	900	580	1043	519	863	824	952	731
FD2 (Hz)	3295	3470	3234	3171	3282	3125	3291	3375	3316	3368

$\frac{1}{2N+1}\sum_{m=-N}^{N} y_2[n+m]$. Where, 2N+1 is the window length in terms of sample number. The resultant signal is called the ZFF signal. Its positive giving zero crossings indicate the glottal closure instants (GCIs), which are used to estimate the F0 (Murty and Yegnanarayana, 2008).

The excitation feature, SoE was derived using the ZFF method. The slope of the ZFF signal around the glottal closure instants (GCIs) gives a measure of the SoE (Murty and Yegnanarayana, 2008; Murty et al., 2009; Mittal and Yegnanarayana, 2015b).

3.2 Vocal Tract Filter Features

The first five formants (F1 to F5) were derived by using linear prediction (LP) spectrum (Makhoul, 1975; Hermansky, 1990; Atal and Hanauer, 1971; Yegnanarayana, 1978). The sound files were resampled to 10 KHz and LP order as 10.

The first two dominant peak frequencies (FD1 and FD2) were derived from the acoustic signal using LP analysis (Makhoul, 1975; Hermansky, 1990). With the LP order 5, the LP spectrum will have a maximum of two peaks corresponding to two complex conjugate pole pairs (Mittal et al., 2014). The corresponding frequencies of these two peaks are known as the dominant frequencies, denoted as FD1 and FD2, respectively (Mittal and Yegnanarayana, 2015a). The dominant frequencies represent the frequency response with high spectral energies. These high spectral energies give an idea of the concentration of energy in the spectrum (Mittal and Yegnanarayana, 2015a).

3.3 Combined Features

The E (Rihaczek, 1968) was calculated using the frame size 30 ms and frame shift 10 ms. Signal energy of a discrete-time signal $x[n]$ can be computed as $E_w = \sum_{n=-w/2}^{w/2} |x[n]|^2$. Where, w is the window length.

In the context of discrete-time signals, ZCR is defined as the number of times in any specific time interval/frame that the amplitude of the speech signal goes through a value of zero (Bachu et al., 2008). The definition of ZCR as given in (Bachu et al., 2008) is $Z_n = \sum_{m=-\infty}^{\infty} |sgn[x(m)] - sgn[x(m-1)]|\, w(n-m)$. Where, $sgn[x(n)] = \begin{cases} 1, x(n) \geq 0 \\ -1, x(n) < 0 \end{cases}$ and $w(n) = \begin{cases} \frac{1}{2N} for, 0 \leq n \leq N-1 \\ 0\, for, otherwise \end{cases}$.

4 Results and Observations

The obtained results indicate higher μ_{F0} values for the children with ASD as compared with the normal children, and this statement is true for all English vowels. Besides, according to the tongue position, female children with ASD have the highest μ_{F0} value for mid-vowel /e/ and have the low-

Table 3: Mean (μ) Values of the Source Features (*F0* and *SoE*), Combined Source-filter Features (*Energy E* and *Zero-crossing Rate ZCR*) and Vocal Tract Filter Features (*Formants Frequencies* and *Dominant Frequencies*) of the *Female* Children with *ASD* and *Normal* (Nm): (a) Acoustic Features and (b)-(f) Mean Values for Five English Vowels; *F1* to *F5* Indicate First Five *Formants Frequencies*, Respectively, and *FD1* and *FD2* are First and Second *Dominant Frequencies*, Respectively

(a) Features	*(b) /a/*		*(c) /e/*		*(d) /i/*		*(e) /o/*		*(f) /u/*	
	ASD	*Nm*	*ASD*	*Nm*	*ASD*	*Nm*	*ASD*	*Nm*	*ASD*	*Nm*
F0 (Hz)	326	314	343	321	339	330	340	310	335	313
SoE×100	32.0	30.5	37.1	41.9	38.1	50.9	42.9	33.0	37.1	35.3
E×1000	39.5	23.4	35.3	24.2	48.3	20.7	63.7	34.9	58.6	30.3
ZCR×1000	35.6	55.3	26.9	48.3	29.3	39.4	29.9	38.3	32.1	40.8
F1 (Hz)	711	457	572	517	633	438	670	646	693	546
F2 (Hz)	1554	1261	1636	1213	1630	1141	1322	1231	1438	1278
F3 (Hz)	2653	2484	2776	2523	2746	2476	2537	2487	2588	2532
F4 (Hz)	3720	3559	3778	3571	3782	3501	3558	3612	3629	3649
F5 (Hz)	4439	4411	4425	4404	4429	4411	4417	4345	4396	4348
FD1 (Hz)	865	827	686	783	681	560	803	784	860	810
FD2 (Hz)	3185	3436	3286	3111	3269	3129	3058	3432	3112	3175

est μ_{F0} value for low-vowel /a/ as compared with other English vowels. But, in the case of the normal female children, high-vowel /i/ gives the highest and mid-vowel /o/ gives the lowest μ_{F0} values as compared with other English vowels. However, in the case of the male children with ASD, such results have not been found. It is observed that male children with ASD follow a similar μ_{F0} trend with the normal male children for all English vowels. These results can be analyzed from Table 2 and 3.

Like μ_{F0}, in the case of μ_E also, the children with ASD have higher values for all the five English vowels as compared with the normal children. Also, for all the five English vowels, the female children with ASD have higher μ_E values as compared with the male children with ASD, but this is vice versa for the normal children. Besides, in the case of the children with ASD, the same vowel /e/ has the lowest μ_E values for both male and female children, whereas this is not the same for the normal male and female children. Likewise, in the case of the normal children, the same vowel /o/ has the highest μ_E values for both male and female children, whereas this is not true for the male and female children with ASD. These statements can be observed from μ_E values in Table 2 and 3.

Regarding μ_{SoE}, only front vowels /e/ and /i/ indicate lower values for the children with ASD as compared with the normal children. But, in the case of mid and rear vowels, i.e., /a/, /o/, and /u/, μ_{SoE} indicate higher values for the children with ASD than the normal children. Besides, in the case of both the normal male and female children, the same vowel /i/ has the highest μ_{SoE} values as compared with other English vowels. But, this statement is not true in the case of the children with ASD. Again, in the case of both the male and female children with ASD, the same vowel /a/ has the lowest μ_{SoE} values as compared with other English vowels, whereas this is not the case with the normal children. All these results can be observed from μ_{SoE} values, tabulated in Table 2 and Table 3.

The μ_{ZCR} have lower values for the children with ASD as compared with the normal children, and it is true for all English vowels. This observation is graphically represented in Figure 1(g) and 1(h). Also, in the case of the front and mid vowels, i.e., /a/, /e/, and /i/, the male children with ASD have higher μ_{ZCR} values as compared with the female. But, it is vice versa in the case of the normal children. Besides, in the case of both male and female children with ASD, the same vowel /e/ has the lowest μ_{ZCR} values as compared with other English vowels, whereas this is not the case with the normal children. These results can be observed from μ_{ZCR} values, given in Table 2 and 3.

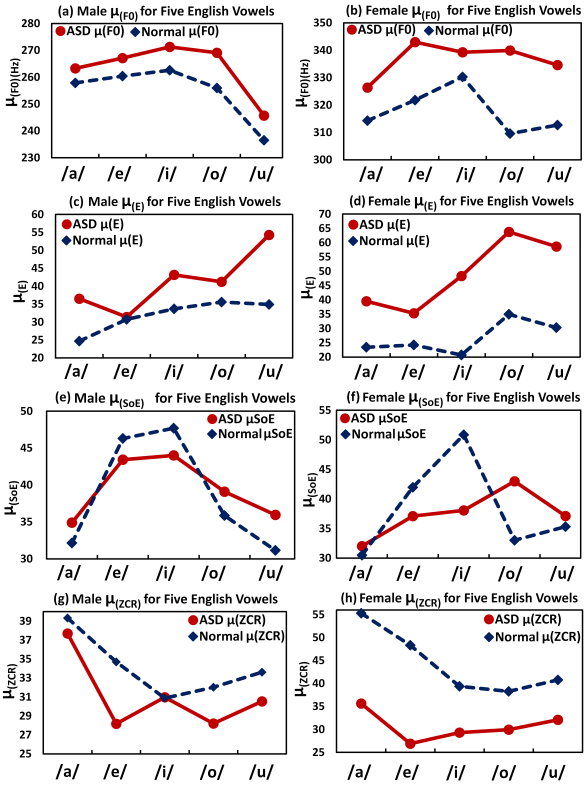


Figure 1: Differences in the Mean Values of F0, E, SoE, and ZCR between the *ASD Affected* and the *Normal* Children.

The children with ASD have significantly higher μ_{F1} values for all English vowels as compared with the normal children. Next, it is observed that the normal female children have higher μ_{F1} values for all the five English vowels as compared with the normal male children, whereas this statement is not true in the case of the children with ASD. According to the tongue position, in the case of both the male and female children with ASD, the μ_{F1} indicates the highest values for the low vowel /a/ as compared with the high and mid vowels. But, in the case of both the male and female normal children, the μ_{F1} indicates the highest values for the mid vowel /o/ as compared with the high and low vowels. The μ_{F1} results are tabulated in Table 2 and 3.

The μ_{F2} values are higher for all English vowels in the case of the children with ASD as compared with the normal children. Also, the μ_{F2} values for all the five English vowels of both the male and female children with ASD follow a similar trend, whereas there is no such trend observed in the case of the normal children. Besides, according to the tongue position, both the male and female children with ASD have the highest μ_{F2} values for the mid vowel /e/ as compared with the high and low vowels. But, in the case of the normal children, as compared with the mid and low vowels the high vowels /i/ and /u/ give the highest μ_{F2} values for both the male and female children, respectively. All these results can be analyzed from μ_{F2} values tabulated in Table 2 and 3.

Like μ_{F1} and μ_{F2}, the μ_{F3} values are also higher for all English vowels in the case of the children with ASD as compared with the normal children. According to the tongue position, in the case of both the male and female children with ASD, the μ_{F3} indicates the highest values for the mid vowel /e/ as compared with the high and low vowels. But, in the case of the normal children, the μ_{F3} indicates the highest values for the high vowels (/i/ and /u/) as compared with the mid and low vowels. The μ_{F3} values are tabulated in Table 2 and 3.

As compared with the normal children, the children with ASD have higher μ_{F4} values for the front and mid vowels only. Next, according to the tongue position, in the case of both the male and female children with ASD, the μ_{F4} gives the highest values for the mid vowel /o/ as compared with the high and low vowels. But, this is not the case for the normal children. The μ_{F4} results can be analyzed from the Figure 2(g) and 2(h), also from the μ_{F3} values, tabulated in Table 2 and 3.

The μ_{F5} indicates higher values for all the five English vowels in the case of the children with ASD as compared with the normal children, depicted in Figure 2(i) and 2(j). Also, both the male and female normal children have the lowest μ_{F5} values for the mid vowel /o/ as compared with the high and low vowels. But, this statement is not true in the case of the ASD children. The μ_{F5} values are tabulated in Table 2 and 3.

All the five English vowels have higher μ_{FD1} values for the children with ASD as compared with the normal children, depicted in Figure 3(a) and 3(b). According to the tongue position, both the male and female normal children have the lowest μ_{FD1} values for the high vowel /i/ as compared with the mid and low vowels. But, in the case of the ASD children, as compared with the high and low vowels the mid vowels /e/ and /o/ indicate the lowest μ_{FD1} values for both the female and male, respectively. The μ_{FD2} results can be analyzed from Table 2 and 3.

In the case of μ_{FD2}, only the front vowel /e/ and

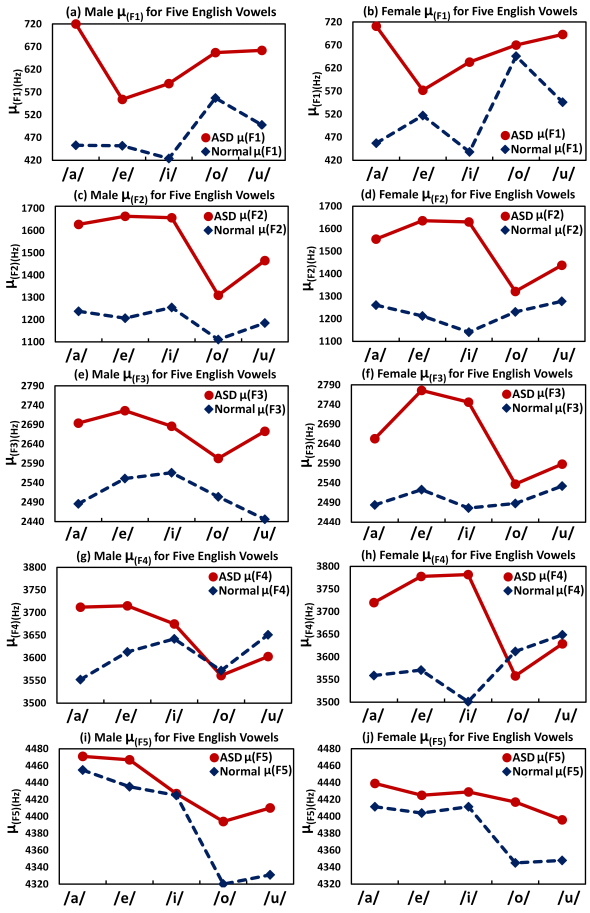


Figure 2: Differences in the Mean Values of *Formants Frequencies* (F1, F2, F3, F4, and F5) between the *ASD Affected* and the *Normal* Children.

/i/ have higher values for the children with ASD as compared with the normal children, graphically shown in Figure 3(c) and 3(d). In addition, according to the tongue position, both the male and female normal children have the highest μ_{FD2} values for the low vowel /a/ as compared with the mid and high vowels. On the other hand, as compared with other English vowels the high vowel /u/ has the highest μ_{FD2} value for the male ASD group and the mid vowel /e/ has the highest μ_{FD2} value for the female ASD group. The μ_{FD2} values are tabulated in Table 2 and 3 for the male and female children, respectively.

5 Analyses of Results

This section describes the observed results in speech production point of view. Firstly, the F0 which reveals the source characteristics of the speech production system, the result infers that in the case of all the five English vowels, the male and female children with ASD have a higher vocal fold vibration rate than the normal male and female children. This statement is true for all the five English vowels. Furthermore, in the case of female children with ASD, mid-vowel /e/ has the highest and low-vowel /a/ has the lowest vocal fold vibration rate as compared with other English vowels. On the other hand, in the case of the normal female children, high-vowel /i/ has the highest and mid-vowel /o/ has the lowest vocal fold vibration rate as compared with other English vowels. These observations can be analyzed from Figure 1(a) and 1(b).

In the case of E which gives the information about the combined source-system characteristics of the speech production system, the result implies that the children with ASD have louder speech and put more vocalization effort than the normal children. Also, in the case of all English vowels the female children with ASD put more vocalization effort than the male children with ASD, but this is vice versa in the case of the normal group. These results can be analyzed from μ_E values graphically depicted in Figure 1(c) and 1(d).

The observed SoE result infers that in the case of the front vowels the strength of impulse-like excitation is lower during the glottal activity (vibration of vocal folds) of the children with ASD as compared with the normal children. But, in the case of mid and rear vowels the strength of impulse-like excitation is higher for the ASD children than the normal children. This result can be analyzed from Figure 1(e) and 1(f).

The F1 result implies that in the case of all five English vowels, the children with ASD have a lesser oral constriction in the front half of the oral section of the vocal tract as compared with the normal children. Again, in terms of pharyngeal constriction, it can be stated that during the pronunciation of all the five English vowels the pharyngeal constriction is greater for the children with ASD as compared with the normal children. The F1 observed result also implies that both the male and female children with ASD have the greatest pharyngeal constriction for the low-vowel /a/ as compared with the mid and high vowels. But, the normal male and female children have the greatest pharyngeal constriction for the mid-vowel /o/ as compared with the high and low vowels. Furthermore, during the pronunciation of all English vowels the children with ASD increase their tongue higher than the normal children. Because the F1

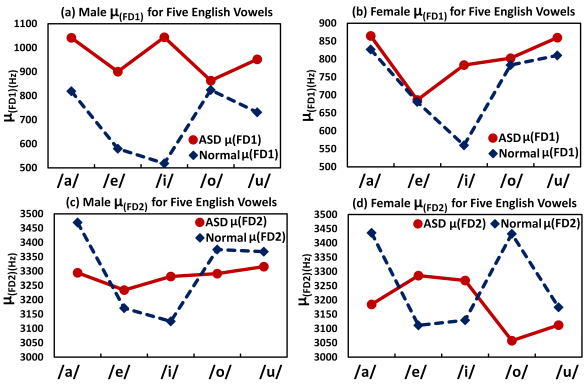


Figure 3: Differences in the Mean Values of *Dominant Frequencies* (FD1 and FD2) between the *ASD Affected* and the *Normal* Children.

value increases with increasing the tongue position higher. The F1 values for all English vowels are graphically depicted in Figure 2(a) and 2(b).

The F2 result implies that in the case of all English vowels the back tongue constriction is lesser and the front tongue constriction is greater for the children with ASD than the normal children. Furthermore, it can be stated from the observed result that both the male and female children with ASD have the least back tongue constriction and the greatest front tongue constriction for the mid vowel /e/ as compared with the high and low vowels. On the other hand, the normal male children have the least back tongue constriction and the greatest front tongue constriction for the high-vowel /i/ as compared with the mid and low vowels, and the normal female children have the least back tongue constriction and the greatest front tongue constriction for the high-vowel /u/ as compared with the mid and low vowels. This observation can be analyzed from Figure 2(c) and 2(d).

The F3 result implies that in the case of the children with ASD lip-rounding is lesser during the pronunciation of all English vowels. Hence, the constriction is least and as a result all English vowels give higher μ_{F3} frequency values for the children with ASD as compared with the normal children. The results are graphically depicted in Figure 2(e) and 2(f).

Also, the results of the first three formants (F1, F2 and F3) indicate that the length of the pharyngeal-oral tract is shorter in the case of the children with ASD as compared with the normal children. Because, the formants values of vowels are inversely proportional to the pharyngeal-oral tract, and here the children with ASD have higher μ_{F1}, μ_{F2} and μ_{F3} values for all English vowels as compared with the normal children. Also, in terms of the lip-rounding, the F1, F2, F3 and F5 results imply that the children with ASD have a lesser lip-rounding as compared with the normal group.

In the case of formants frequencies and dominant frequencies, the differences between the ASD and the normal children are highest for vowel /i/. It implies that ASD children have probably more difficulty in pronouncing the words with vowel /i/.

6 Key Contributions

The key contributions of this study are as follows:

- The ASD and the normal children's speech datasets are collected by recording the speech samples of non-native Indian English speakers.
- Only English vowels (/a/, /e/, /i/, /o/, and /o/) are considered in this study.
- Some of the robust speech features like SoE, F5, FD1, and FD1 are considered here, which were not considered in similar types of previous studies.
- The F0, E, F1, F2, F3, and F5 results clearly distinguish the ASD and the normal children. All these features have significantly higher mean values for all English vowels in the case of the ASD children as compared with the normal children.
- The results of the formants and dominant frequencies indicate that children with ASD have probably more difficulty in pronouncing the words with vowel /i/.

7 Conclusions

The aim of this study is to analyze differences in various speech production features of the children with ASD as compared with the normal children. Only English vowels sounds are used in this study. An autism speech dataset and a normal childrens speech dataset are recorded separately for this research purpose. Then, differences between the children with ASD and the normal children are analyzed by observing the source characteristics (F0 and SoE), system characteristics (dominant frequencies and formants), and combined characteristics (ZCR and E). It is observed that there are significant differences between the ASD and the

normal children, in terms of their speech production characteristics in English vowels regions. In the case of most of the speech production features, the ASD children have significantly higher values than the normal children. These acoustic characteristics of the children with ASD can be used as markers to identify ASD. But, we did not find any single speech feature that can be utilized as a diagnostic marker for ASD.

A small size of speech data for female ASD children is a limitation of this study. In future studies, we will try to find a single speech feature that can be utilized as an acoustic marker to identify ASD.

Acknowledgments

The authors are thankful to Dr. N. P. Karthikeyen and DOAST Integrated Therapy Centre for Autism, Chennai, India, for providing the opportunity for ASD children's voice recording. The authors are also thankful to Chinmaya Vidhyalaya school, Sri City, Andhra Pradesh, India, for providing the opportunity to record normal children's speech dataset.

References

Meysam Asgari, Alireza Bayestehtashk, and Izhak Shafran. 2013. Robust and accurate features for detecting and diagnosing autism spectrum disorders. In *Interspeech*, pages 191–194.

Bishnu S Atal and Suzanne L Hanauer. 1971. Speech analysis and synthesis by linear prediction of the speech wave. *The journal of the acoustical society of America*, 50(2B):637–655.

Autism and Developmental Disabilities Monitoring Network Surveillance Year 2010 Principal Investigators. 2014. Prevalence of autism spectrum disorder among children aged 8 yearsautism and developmental disabilities monitoring network, 11 sites, united states, 2010. *Morbidity and Mortality Weekly Report: Surveillance Summaries*, 63(2):1–21.

RG Bachu, S Kopparthi, B Adapa, and BD Barkana. 2008. Separation of voiced and unvoiced using zero crossing rate and energy of the speech signal. In *American Society for Engineering Education (ASEE) Zone ference Proceedings*, pages 1–7.

Christiane AM Baltaxe. 1977. Pragmatic deficits in the language of autistic adolescents. *Journal of Pediatric Psychology*, 2(4):176–180.

Christiane AM Baltaxe and James Q Simmons. 1985. Prosodic development in normal and autistic children. In *Communication problems in autism*, pages 95–125. Springer.

Daniel Bone, Theodora Chaspari, Kartik Audhkhasi, James Gibson, Andreas Tsiartas, Maarten Van Segbroeck, Ming Li, Sungbok Lee, and Shrikanth Narayanan. 2013. Classifying language-related developmental disorders from speech cues: the promise and the potential confounds. In *INTERSPEECH*, pages 182–186.

Yoram S Bonneh, Yoram Levanon, Omrit Dean-Pardo, Lan Lossos, and Yael Adini. 2011. Abnormal speech spectrum and increased pitch variability in young autistic children. *Frontiers in human neuroscience*, 4:237.

Julie Brisson, Karine Martel, Josette Serres, Sylvain Sirois, and Jean-Louis Adrien. 2014. Acoustic analysis of oral productions of infants later diagnosed with autism and their mother. *Infant mental health journal*, 35(3):285–295.

Anne-Marie R DePape, Aoju Chen, Geoffrey BC Hall, and Laurel J Trainor. 2012. Use of prosody and information structure in high functioning adults with autism in relation to language ability. *Frontiers in psychology*, 3:72.

Joshua J Diehl, Duane Watson, Loisa Bennetto, Joyce McDonough, and Christine Gunlogson. 2009. An acoustic analysis of prosody in high-functioning autism. *Applied Psycholinguistics*, 30(3):385–404.

Riccardo Fusaroli, Anna Lambrechts, Dan Bang, Dermot M Bowler, and Sebastian B Gaigg. 2017. Is voice a marker for autism spectrum disorder? a systematic review and meta-analysis. *Autism Research*, 10(3):384–407.

Hynek Hermansky. 1990. Perceptual linear predictive (plp) analysis of speech. *the Journal of the Acoustical Society of America*, 87(4):1738–1752.

Kathleen Hubbard and Doris A Trauner. 2007. Intonation and emotion in autistic spectrum disorders. *Journal of psycholinguistic research*, 36(2):159–173.

Yasuhiro Kakihara, Tetsuya Takiguchi, Yasuo Ariki, Yasushi Nakai, Satoshi Takada, Y Kakihara, et al. 2015. Investigation of classification using pitch features for children with autism spectrum disorders and typically developing children. *Am. J. Sign. Process*, 5:1–5.

Leo Kanner et al. 1943. Autistic disturbances of affective contact. *Nervous child*, 2(3):217–250.

Margaret M Kjelgaard and Helen Tager-Flusberg. 2001. An investigation of language impairment in autism: Implications for genetic subgroups. *Language and cognitive processes*, 16(2-3):287–308.

Manoj Kumar, Pooja Chebolu, So Hyun Kim, Kassandra Martinez, Catherine Lord, and Shrikanth Narayanan. 2018. A knowledge driven structural segmentation approach for play-talk classification during autism assessment. In *Interspeech*.

Catherine Lord, Michael Rutter, and Ann Le Couteur. 1994. Autism diagnostic interview-revised: a revised version of a diagnostic interview for caregivers of individuals with possible pervasive developmental disorders. *Journal of autism and developmental disorders*, 24(5):659–685.

John Makhoul. 1975. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580.

Erik Marchi, Björn Schuller, Simon Baron-Cohen, Ofer Golan, Sven Bölte, Prerna Arora, and Reinhold Häb-Umbach. 2015. Typicality and emotion in the voice of children with autism spectrum condition: Evidence across three languages. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Joanne McCann and Sue Peppé. 2003. Prosody in autism spectrum disorders: a critical review. *International Journal of Language & Communication Disorders*, 38(4):325–350.

Vinay K Mittal and Bayya Yegnanarayana. 2015a. Analysis of production characteristics of laughter. *Computer Speech & Language*, 30(1):99–115.

Vinay Kumar Mittal and B Yegnanarayana. 2015b. Study of characteristics of aperiodicity in noh voices. *The Journal of the Acoustical Society of America*, 137(6):3411–3421.

Vinay Kumar Mittal, B Yegnanarayana, and Peri Bhaskararao. 2014. Study of the effects of vocal tract constriction on glottal vibration. *The Journal of the Acoustical Society of America*, 136(4):1932–1941.

Emily Mower, Chi-Chun Lee, James Gibson, Theodora Chaspari, Marian E Williams, and Shrikanth Narayanan. 2011. Analyzing the nature of eca interactions in children with autism. In *Twelfth Annual Conference of the International Speech Communication Association*.

K Sri Rama Murty and B Yegnanarayana. 2008. Epoch extraction from speech signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(8):1602–1613.

K Sri Rama Murty, Bayya Yegnanarayana, and M Anand Joseph. 2009. Characterization of glottal activity from speech signals. *IEEE signal processing letters*, 16(6):469–472.

Yasushi Nakai, Ryoichi Takashima, Tetsuya Takiguchi, and Satoshi Takada. 2014. Speech intonation in children with autism spectrum disorder. *Brain and Development*, 36(6):516–522.

D Kimbrough Oller, P Niyogi, S Gray, Jeffrey A Richards, Jill Gilkerson, Daoyi Xu, Umit Yapanel, and Steven F Warren. 2010. Automated vocal analysis of naturalistic recordings from children with autism, language delay, and typical development. *Proceedings of the National Academy of Sciences*, 107(30):13354–13359.

Julia Parish-Morris, Mark Liberman, Neville Ryant, Christopher Cieri, Leila Bateman, Emily Ferguson, and Robert Schultz. 2016. Exploring autism spectrum disorders using hlt. In *Proceedings of the third workshop on computational lingusitics and clinical psychology*, pages 74–84.

Rhea Paul, Lawrence D Shriberg, Jane McSweeny, Domenic Cicchetti, Ami Klin, and Fred Volkmar. 2005. Brief report: Relations between prosodic performance and communication and socialization ratings in high functioning speakers with autism spectrum disorders. *Journal of Autism and Developmental Disorders*, 35(6):861.

Jean Quigley, Sinéad McNally, and Sarah Lawson. 2016. Prosodic patterns in interaction of low-risk and at-risk-of-autism spectrum disorders infants and their mothers at 12 and 18 months. *Language Learning and Development*, 12(3):295–310.

A Rihaczek. 1968. Signal energy distribution in time and frequency. *IEEE Transactions on information Theory*, 14(3):369–374.

Joao F Santos, Nirit Brosh, Tiago H Falk, Lonnie Zwaigenbaum, Susan E Bryson, Wendy Roberts, Isabel M Smith, Peter Szatmari, and Jessica A Brian. 2013. Very early detection of autism spectrum disorders based on acoustic analysis of pre-verbal vocalizations of 18-month old toddlers. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7567–7571. IEEE.

Andrew B Short and Eric Schopler. 1988. Factors relating to age of onset in autism. *Journal of autism and developmental disorders*, 18(2):207–216.

Lawrence D Shriberg, Rhea Paul, Jane L McSweeny, Ami Klin, Donald J Cohen, and Fred R Volkmar. 2001. Speech and prosody characteristics of adolescents and adults with high-functioning autism and asperger syndrome. *Journal of Speech, Language, and Hearing Research*, 44(5):1097–1115.

Helen Tager-Flusberg, Rhea Paul, Catherine Lord, F Volkmar, Rhea Paul, and Ami Klin. 2005. Language and communication in autism. *Handbook of autism and pervasive developmental disorders*, 1:335–364.

Lorna Wing, Judith Gould, and Christopher Gillberg. 2011. Autism spectrum disorders in the dsm-v: better or worse than the dsm-iv? *Research in developmental disabilities*, 32(2):768–773.

B Yegnanarayana and K Sri Rama Murty. 2009. Event-based instantaneous fundamental frequency estimation from speech signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):614–624.

Bayya Yegnanarayana. 1978. Formant extraction from linear-prediction phase spectra. *The Journal of the Acoustical Society of America*, 63(5):1638–1640.

A Survey on Ontology Enrichment from Text

Vivek Iyer
IIIT, Gachibowli
Hyderabad - 500032

Lalit Mohan
IIIT, Gachibowli
Hyderabad - 500032

Y. Raghu Reddy
IIIT, Gachibowli
Hyderabad - 500032

Mehar Bhatia
Shiv Nadar University
Greater Noida
India - 201314

Abstract

Increased internet bandwidth at low cost is leading to the creation of large volumes of unstructured data. This data explosion opens up opportunities for the creation of a variety of data-driven intelligent systems, such as the Semantic Web. Ontologies form one of the most crucial layers of semantic web, and the extraction and enrichment of ontologies given this data explosion becomes an inevitable research problem. In this paper, we survey the literature on semi-automatic and automatic ontology extraction and enrichment and classify them into four broad categories based on the approach. Then, we proceed to narrow down four algorithms from each of these categories, implement and analytically compare them based on parameters like context relevance, efficiency and precision. Lastly, we propose a Long Short Term Memory Networks (LSTM) based deep learning approach to try and overcome the gaps identified in these approaches.

1 Introduction

There has been an explosion of data on the Internet in the past few years, primarily caused by the drastic increase in the number of internet users over the years. About 90% of the data on internet has been created since 2016, mainly because of the massive increase in the user base and machine to machine communication. Data is defined as unprocessed facts and figures that do not contain any added interpretation or analysis. Information is interpretation of structured or unstructured data so that it holds meaning. Knowledge is processed information, experience, and insight combined such that it is beneficial to the end user[1].

Web pages, the primary source of *knowledge* on the World Wide Web (WWW) are primarily text documents annotated using Hypertext Markup Language (HTML). Lack of semantic markup of pages can result in irrelevant search results. The semantic web[2] provides a format or structure to machines to understand the *meaning* of the web page data rather relying on HTML markup, to make web intelligent and intuitive to user's queries. The semantic web includes data-centric publishing languages, including RDF (Resource Description Framework - the data modeling language for the semantic web), SPARQL (SPARQL protocol and RDF query language for semantic web) and OWL (Web Ontology Language - schema language, or knowledge representation language, of the semantic web), which allows meaning and structure to be added to content in a machine-readable format. OWL [3] allows definition of concepts composably, i.e. in such a way that it allows the reuse of concepts and relationships. Given the amount of information being extracted from the data generated on a regular basis in various domains, it becomes essential for it to be stored in the form of knowledge in ontologies. However, the knowledge stored in ontologies is rarely static. Like all other knowledge structures, its vital for ontologies to be enriched with time so as to improve the quality of search results.

Given recent advances in the fields of artificial intelligence and machine learning, as well as increased data processing capabilities with increase in compute power, newer, better and more accurate ways of extracting and enriching ontologies from text are now possible. Ontology extraction from text has primarily been at lower layers in ontology "layer cake" (Buitelaar et al.,

[1] https://tinyurl.com/datainfknowledge

[2] https://expertsystem.com/what-is-the-semantic-web/

[3] https://db-x.org/blog/2016/04/15/semantic-web-2/

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 95–104
Hyderabad, India, December 2019.

2005). A pre-existing seed ontology created manually or through learning needs enrichment. Ontology enrichment (Faatz and Steinmetz, 2002) is population, updation, and adaptation (Noy and Klein, 2004) of concepts, relations and rules. In the context of this paper, we assume a pre-existing seed ontology that is enriched by learning (semi-automatic or automatic) from text. The survey in this paper attempts to address the following important research questions:

- How are ontologies enriched by learning from unstructured text, and which algorithms are considered seminal?
- How do seminal algorithms compare with each other, in regards to context relevance, algorithmic efficiency and precision?
- What gaps are identified in these algorithms, and how can they be potentially addressed?

We did not focus on the other knowledge representation methods such as knowledge graphs, frames, semantic nets and others in the survey considering the extensivity of ontology research and the generalizability of research trends to other knowledge representation methods. The further sections of the document contain our literature survey approach for identifying the state-of-the-art in section 2; we explain the broad genres identified in the ontology extraction in section 3; we proceed with a critical analysis of the major approaches through the years, by analyzing the algorithms on context relevance, efficiency and precision in section 4; we propose a deep learning based methodology (LSTM - Long Short Term Memory) to possibly overcome the gaps in the ontology enrichment in section 5 and finally end with a conclusion summarizing our observations.

2 Approach for Literature Review

We started the review on ontology learning from text before focusing on enrichment. Research on ontology learning from text started in 1995 (Mahesh et al., 1995) but still continues to be an area of interest. In the last two decades, there have been 20 survey papers on ontology tools, learning, evolution, construction, enrichment, change, generation, population, and matching with text. The large count of survey papers indicates the growing interest among researchers and changing research approaches in ontology learning. We classified these survey papers[4] on the basis of text format (structured or unstructured), evaluation methods, ontology layer cake, AI techniques, level of automation, etc. Most survey papers recommended human intervention, continued automation, gold standards and graphical interfaces for improved quality, expressiveness and scalability. While the survey papers were thorough, there weren't any papers that follow the systematic literature review (SLR) or systematic mapping process (Kitchenham, 2004), or any that discussed seminal papers that led to change of approaches.

Based on our study of the survey papers' classification methods and future directions, the keywords for search from digital libraries were "Extraction", "Evolution", "Enrichment", "Maintain", and "Learning" along with "Ontology" as keyword. We did not follow SLR process as our objective was to analyze the seminal papers based on context relevance, precision and algorithm efficiency. The input to our survey process consisted of 166 research papers extracted from ScienceDirect, Springer, IEEE, and ACM digital libraries from 1990-2018 time period. After reviewing the abstract and conclusion, 65 papers were eliminated from the list as they were thesis, patents, grey material, non-English, position or tutorial papers and others. The papers related to construction of data, text summarization using ontologies, machine translation, Information Retrieval, etc, of the extracted research papers were also excluded from further analysis. While there were about 23 domains for validation, Medical and Education domains were the most referred domains in the shortlisted papers. The Figure 1 (Y-axis is the count of papers and X-axis is the year of publication) on ontology learning depicts the ongoing interest of researchers. The study on approaches of the shortlisted papers stated that although natural language processing and description logic continue to be used; Word2Vec, a step towards deep learning is being more leveraged for ontology learning. The shortlisted papers were categorized after reading the abstract, introduction and conclusion, as shown in Figure 2. The papers on "create" were related to ontology construction or population or generation. The papers on "update" were related

[4]https://tinyurl.com/OntoSurvey

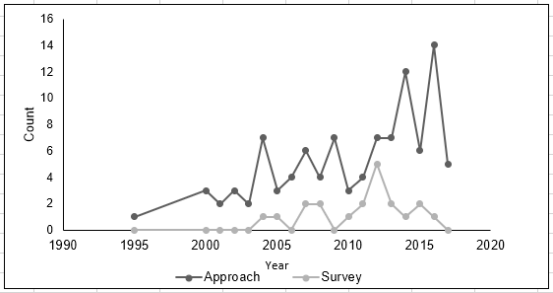


Figure 1: Trend Chart on Ontology Research

to ontology evolution, enrichment, updation, refinement, maintain, etc. The papers on "CRUD" operation dealt with creation, updation and deletion of redundant concepts and relations as well. For further analysis, the papers on "update" and "CRUD" on ontology were clustered into 4 categories based on the approach used for enrichment.

1. Similarity Based Clustering Algorithms
2. Set Theoretic Based Algorithms
3. Web Corpus Based Algorithms
4. Deep Learning Based Algorithms

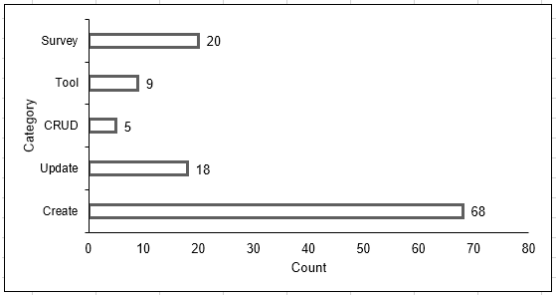


Figure 2: Ontology Learning Categories

3 Categories in Ontology Enrichment

We proceeded with a review of the 23 shortlisted papers of the 4 categories in ontology enrichment.

3.1 Similarity Based Clustering Algorithms:

Some of the earliest papers in the field of ontology enrichment from text, adopted similarity-based clustering approach. A hierarchical clustering algorithm to classify ontology-based metadata (Maedche and Zacharias, 2002) was proposed in 2002. Later, a similarity-based clustering approach was proposed to identify concepts in a gene ontology (Cheng et al., 2004). The unsupervised guided hierarchical clustering algorithm (Cimiano and Staab, 2005) uses an oracle of hypernyms derived from WordNet, text and WWW corpora for clustering concepts in a hierarchy. The fuzzy inference mechanism (Lee et al., 2007) uses fuzzy numbers that calculate the conceptual similarity between concepts to obtain new learning instances.

3.2 Set Theoretic Based Algorithms

These algorithms used a set-theoretic approach to order concepts. Harris's distributional hypothesis (Sahlgren, 2008) modeled the context of a certain word with its dependencies, and on the basis of this information, Formal Concept Analysis (FCA) (Cimiano et al., 2005a) outputs a concept lattice which is then converted into a concept hierarchy. Also, algorithms and transformations that combine FCA and the Horn model (Ben-Khalifa and Motameny, 2007) of a concept lattice have been proposed (Haav, 2004). A fuzzy extension of FCA (De Maio et al., 2009) described an approach for automatic elicitation of ontologies by web analysis. It also formalized a method that generated an OWL-based representation of concepts, individuals and properties.

Relational Concept Analysis (RCA) (Hacene et al., 2008) constructs ontologies in a semi-automated manner by translating concept lattices with interrelated elements to concepts and relations in the ontology. RCA is an extension of FCA that allows for the processing of multi-relational datasets, each with its own set of attributes and relationships amongst themselves.

3.3 Web Corpus Based Algorithms

Web corpus Based Algorithms used web as a big data corpus to overcome problems of data sparsity. The categories and labels from Wikipedia were used to classify concepts (Cui et al., 2009; Ahmed et al., 2012; Medelyan et al., 2009) leveraging N-grams and other related NLP algorithms. The Open Linked Data (Booshehri and Luksch, 2014), a freely available source of semantic knowledge is used as a skeleton to construct ontologies (Tiddi et al., 2012). DBPedia, another crowd-sourced Linked Data dataset that extracts structured information from Wikipedia is used to enrich ontology (Booshehri and Luksch, 2015).

An automatic and unsupervised methodology that uses the Web to learn ontological concept properties, or attributes, and attribute restrictions,

was proposed (Sánchez, 2010). In the "Self Annotating Web", globally available knowledge, or syntactic resources, were used for the creation of metadata, the basic idea being that the statistical distribution of syntactic structures on the web can be used to approximate semantics. One such algorithm that implemented this paradigm is called PANKOW (Pattern-based Annotation through Knowledge On the Web) (Cimiano et al., 2004), in which patterns were instantiated from schemata and the number of hits of related entities for each concept were counted. C-PANKOW (Cimiano et al., 2005b), or Context-driven PANKOW that outperforms its predecessor, PANKOW by downloading abstracts offline, performing linguistic analysis and using the context to resolve ambiguity.

3.4 Learning based Algorithms

In recent years, learning algorithms driven by feedback from domain experts have gained popularity. OntoAMAS (Benomrane et al., 2016) tool is based on adaptive multi-agent system (AMAS) for ontology enrichment and makes proposals based on ontologists' feedback. Also noteworth, is the Probabilistic Relational Hierarchy Extraction technique based on Probabilistic Relational Concept Extraction (Drumond and Girardi, 2010) to extract concepts and the taxonomic relationships from inference on Markov Logic Networks. Group storytelling technique has been used (Confort et al., 2015) to gather knowledge from those involved in the field in the first phase, which makes the system learn the concepts for an ontology automatically. OntoHarvester system (Mousavi et al., 2014) used deep NLP-based algorithms to mine text and extract domain-specific ontologies by iteratively extracting ontological relations that link the concepts in the ontology to the terms in the text, out of which strongly connected concepts were added to the ontology.

The Automated Ontology Generation Framework (Alobaidi et al., 2018), used Linked Biomedical Ontologies, various NLP techniques (in text processing based on "Compute on Demand" method, N-Grams, ontology linking and classification), semantic enrichment (using RDF mining), syntactic pattern and graph-based techniques (to extract relations), and domain inference engine (to build the formal ontology). They also proposed Linked Biomedical Ontologies as a promising solution towards automating the ontology generation process in the disease-drug domain. Word2Vec was used (Wohlgenannt and Minic, 2016) to extract similar meaning terms or concepts and to get certain semantic and syntactic relations based on simple vector operations. The word representations derived from traditional Distributional Semantic Models such as Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA) assume that words in similar contexts have similar embeddings. Word embeddings using neural language models, for example, CBOW and Skip gram, begin usage of deep learning. (Casteleiro et al., 2016) focused on the performance of LDA, LSA, Skip gram and CBOW algorithms in ontology enrichment.

4 From Clustering to Learning algorithms: An in-depth analysis

After categorizing our research set of 23 papers into 4 categories, the seminal algorithms from each category are listed below:

1. Similarity Based Clustering Algorithms: Guided Agglomerative Clustering (Cimiano and Staab, 2005)
2. Web Corpus Based Algorithms: C-PANKOW Algorithm (Cimiano et al., 2005b)
3. Set Theoretic Based Algorithms: Constructing a concept hierarchy using Formal Concept Analysis (Cimiano et al., 2005a)
4. Deep Learning Based Algorithms: The Word2Vec-based algorithm (Wohlgenannt and Minic, 2016)

In this section, we performed an in-depth analysis of these algorithms and compared their performance based on ontology evaluation (Netzer et al., 2009) methods like contextual relevance, precision and algorithmic efficiency.

4.1 Guided Agglomerative Clustering

The guided agglomerative clustering algorithm (Cimiano and Staab, 2005) has a citation count of 99 and published in 2005. The paper is based on Harris's distributional hypothesis and works by clustering concepts based on their similarities. Hypernym oracle extracted using different

methods is the driving factor in the clustering process. Hypernyms oracle is constructed with

Hearst1: *NP* such as {*NP*,}* {(and | or)} *NP*
Hearst2: such *NP* as {*NP*,}* {(and | or)} *NP*
Hearst3: *NP* {,*NP*}* {,} or other *NP*
Hearst4: *NP* {,*NP*}* {,} and other *NP*
Hearst5: *NP* including {*NP*,}* *NP* {(and | or)} *NP*
Hearst6: *NP* especially {*NP*,}* {(and|or)} *NP*

Figure 3: Hearst Patterns (Cimiano and Staab, 2005)

the help of Hearst Patterns (Hearst, 1992). Hearst Patterns 3, used Noun Phrases (NPs) consisting of a determiner, an optional adjective sequence and a common noun sequence which constitutes the NP head. The hypernym oracle H(t) is constructed using the following three sources:

1. WordNet: Uses synsets from WordNet for extracting hypernyms
2. Text Corpus using Hearst Patterns: Hearst Patterns were matched against the underlying text corpus, by using a regular expression comprising of POS tags to match Noun Phrases, thus constructing an is-a relation between the two terms.
3. WWW Corpus using Hearst Patterns: Every concept of interest is instantiated in a Hearst Pattern to form queries to Google API, and the abstracts from the results were downloaded offline. Hearst patterns were matched against these abstracts similar to how they were matched in the text corpus, and is-a relations were extracted accordingly.

The algorithm takes a list of words to be clustered as input. Once the hypernym oracle was constructed, each of these terms were paired up and sorted in the descending order of similarity. The clustering algorithm used the oracle to construct parent-child or sibling relationships between these terms. After this step, the unclassified terms were classified using the r-matches relation.

Though WordNet provides easy and accurate hypernyms, it is not extensive and has a very limited scope. It does not classify proper nouns or infrequently occurring terms, leading to most instances remaining unclassified leading to sparsity and scalability issues. Moreover, matching Hearst Patterns had very bad precision (13%), as shown in Figure 4 and outputs a lot of noisy data. This is due to the algorithm paying no attention to context relevance and extracting hypernyms that were irrelevant to a domain. The same word that had different meanings in different contexts (for instance, bank - which could refer to a river bank or a blood bank or a financial bank) were clustered together. In addition, this approach disregarded a lot of relevant relations because it relied on an exact syntactic pattern match that pays no attention to semantics.

4.2 C-PANKOW algorithm

The C-PANKOW algorithm (Cimiano et al., 2005b) again by Cimiano et al. has a citation count of 246. The algorithm was based on the paradigm of ”Learning by Googling”. In this paradigm, given an instance, evidence was collected from the internet for the possible concepts. Then, either the instance was mapped to the concept with maximum evidence, or alternatively, an engineer with domain-specific knowledge does mapping manually. The PANKOW (Pattern-based Annotation through Knowledge on the Web) algorithm (Cimiano et al., 2004), the predecessor of the C-PANKOW algorithm, instantiated a query using pre-defined patterns or regular expressions. A one-to-one mapping was done between each concept and instance to generate a query from these patterns. This query, similar to how the hypernym oracle was extracted using the WWW corpus in Guided Clustering, was made available to the Google API and the number of hits for this query were counted. Based on the statistical web fingerprint, or the total number of search results for each entity, the instance were mapped to the concept to get *disambiguation by maximal evidence*. The statistical web fingerprint were presented to the knowledge engineer to review and take the final decision. However, PANKOW had a few disadvantages. Firstly, it issued a large number of requests to the Google Web API, which is proportional to the number of ontology concepts, so it does not scale well for large ontologies. Also, because of the restrictions inherent in the generation of patterns, many actual instances were not found.

C-PANKOW addresses some issues by downloading results of queries, or the abstracts, and then doing the pattern matching locally by linguistic analysis. Downloading web pages

reduced the number of requests made to Google Web API and the network traffic by issuing a constant number of queries per instance. In addition, it factors context into consideration and calculates the contextual similarity between two pages before doing concept-instance mapping, which reduced ambiguity especially in cases where a word has multiple meanings and its meaning depends on context. C-PANKOW presented a novel idea to concept extraction by combining the approaches of maximum frequency-based mapping and document similarity-based filtering. Frequency-based mapping reduces noise and gives only the most relevant relations, whereas similarity-based filtering using Doc2Vec (Lau and Baldwin, 2016) helps partially address the issue of context relevance by preemptively filtering out irrelevant abstracts. These two approaches augmented C-PANKOW's precision (36%) to be more than that of Guided Clustering. The filtering also increased algorithmic efficiency as, unlike Guided Clustering, it does not look for matches in irrelevant documents. However, despite its advantages, since C-PANKOW (like Guided Clustering) uses naive syntactic pattern matching to extract hypernymy relations, it does yield noisy data as well, whilst ignoring relevant results. This is because: a) The pattern matching fails to take semantics and language structure into consideration. b) It is also ineffective in situations where the concept being referred were already defined in an earlier sentence c) Though Doc2Vec does partially address the issue of context relevance at the document level, it does not check the relevance at the sentence or paragraph level, resulting in noisy data as well.

To address the concerns of disambiguation in concepts or relations, agent based models have been proposed for the enrichment of ontologies (Sellami et al., 2013). An agent has local knowledge about itself and other neighbour agents, as well as about the lexical terms and concepts extracted from the corpus. It uses this knowledge to evaluate its own relevance in the ontology and manage its relationships with other agents. When new documents are added to the corpus, or when the ontologist suggests changes to the ontology proposed by the MAS, there were perturbations or disturbances caused in the system. Each agent in the MAS reacts to these perturbations by modifying its relations with other agents, updating its knowledge on and/or communicating with other agents in order to reach a stable state. On reaching this stable state, the MAS proposes a new version of the ontology which is once again presented to the ontologist. The ontologist suggests changes again and this whole process continues iteratively till the MAS reaches a final state where the ontology is not challenged by him anymore. In DYNAMO-MAS (Sellami et al., 2013) word disambiguation is handled by the Teminological Ontological Resource (TOR) model which comprises of a conceptual component (the ontology) and a lexical component (the terminology). Terms were attached to concepts by denotation links and contain a confidence score. These denotation links can be changed by the agents if a request with a higher confidence score is made. Thus, any term is attached by a denotation link to the concept with the highest confidence score. Since the same term can have different meanings in different context, the TOR model is able to disambiguate the meaning using these confidence scores. However, the confidence score is partly generated from a pattern score, which in turn has to be manually defined from empirical evaluations. Moreover, the ontologist has to manually verify the annotations proposed by the MAS which in turn means the text corpus has to be limited to a few hundred documents and cannot work on the larger web corpus. Thus while this approach makes a massive progress towards solving the issue of context relevance, it suffers from scalability issues.

4.3 Constructing concepts using FCA

(Cimiano et al., 2005a) has 693 citations and is the primary source for research on FCA from text corpora. The algorithm was based on Set-Theoretic approach that uses FCA to convert a partial order to a concept hierarchy on the basis of syntactic dependencies taken as features. With NLTK, the Part of Speech (POS) tags are extracted, separated into chunks, reduced to base-form (lemmatized), smoothed to overcome data sparseness, weighted, and only those terms with values above a threshold are converted into a formal context (Ganter and Wille, 1999). FCA (Ganter and Wille, 1996) is then applied to this context to transform into a partial order, which is

then compacted to remove abstract concepts and get the final concept hierarchy. This algorithmic approach uses pseudo-syntactic dependencies to extract concepts from the parse tree. Hence, it significantly outperforms Guided Clustering and C-PANKOW in terms of precision. This algorithm forms clusters and also provides an intentional description for them, leading to better understanding. However, this algorithm does not identify labels that describe the intention of a specific cluster, resulting in sparsely populated concepts. In addition, it is inefficient as construction of a separate concept lattice for every document is time expensive. Thus while it is more efficient than Clustering, it loses out to C-PANKOW in efficiency. However, the greater precision does shows that enriching contextual features using pseudo-syntactic dependencies is a viable alternative that outperforms enriching from parse trees.

4.4 Word2Vec-based algorithm

Word2Vec, a 2-layer neural network has also been used to build a sample ontology learning system (Wohlgenannt and Minic, 2016). The neural nets are trained on the linguistic context by Word2Vec, using two methods: Continuous Bag of Words (CBoW) and skip grams. CBoW is used to predict the context of a word, given the word, while skip grams predict the context given the word as input. Word2Vec allows vector operations, and is trained to output high quality similar terms given any input term. The Word2Vec model can be trained on the Google News corpus on any other large corpus.

The algorithm provided higher percentage of relevant concepts that can be used to enrich the ontology. In addition to having greater precision (60%) and efficiency than the previous algorithms, this algorithm makes headway in solving the issue of contextual relevance by using CBoW and Skip Grams to train the model. However, it does have a few drawbacks. Firstly, for terms that aren't encountered by the model in training corpus, a word embedding is not constructed, hence, concepts remaining unclustered. Secondly, Word2Vec doesn't have any shared representations at sub-word levels. It represents each word as an independent vector, though there could be morphologically similar terms. It also detects concepts that are too close to the original term, like plurals and synonyms which are unneccessarily added to the ontology as separate concepts. Lastly, it necessitates manual intervention after every iteration, unlike the previous algorithms, which in turn means it suffers from scalability issues.

5 Discussion

The Guided Agglomerative Clustering algorithms used Wordnet and Hearst Patterns on corpora to build its hypernym oracle. While Wordnet is able to provide hypernyms for common nouns, it cannot handle proper nouns and phrases, which are often the primary focus while enriching domain specific ontologies. Using Hearst Patterns is inefficient too and results in a lot of noise, due to pattern being matching being purely syntactic with no attention paid to context. Though C-PANKOW is able to improve on precision, efficiency and also partially address the issue of context relevance (using a mixture of frequency-based mapping and document similarity scores), it uses naive syntactic pattern matching which results in selecting irrelevant terms and dropping relevant ones. The DYNAMO-MAS algorithm, despite solving disambiguation and having better precision, has serious limitations like data sparsity and unscalability. FCA, which uses pseudo-syntactic dependencies, was found to have better precision than both Clustering and C-PANKOW. But construction of a concept hierarchy is time inefficient, which is where it loses out to C-PANKOW. The Word2Vec algorithm was able to improve the problems of efficiency, precision and data sparsity by using word embeddings and skip-grams, and was found to outperform previously mentioned algorithms. However, this algorithm also suffers from some shortcomings like the inability to handle previously unencountered words, selecting of too similar terms, scalability issues due to manual intervention etc. We used the 'Information Security' ontology (Ekelhart et al., 2006) based on ISO 27001 for comparing the algorithms. Figure 4 shows comparison of the metrics across these algorithms. All these algorithms have an area of improvement when the current concept and its pronouns are being extracted from text. In the previous approaches, the attributes and relations were mapped to the *pronouns* and not

Algorithm	Precision for		Efficiency	Contextual Relevance
	"Vulnerability"	"Threats"		
Guided Clustering	14%	12%	Slow and time consuming due to pattern matching without any filtering of abstracts	Has no frequency-based/context-based filtering
C-PANKOW	38%	35%	Better performance than Clustering as it filters abstracts and substitutes the construction of parse trees with maximum frequency mapping. Still uses naive pattern matching though	Context is given importance by considering semantic similarity
Formal Concept Analysis	43%	44%	Compartively better than Guided Clustering but slower than C-PANKOW as it requires additional processing for the formation of clusters	* Uses pseudo syntactic dependencies which greatly outperform shallow parsing techniques * Forms clusters and provides an intentional description for these clusters helping in better understanding * Results in sparsely populated concepts due to inability to recognise labels for these clusters
Word2Vec	60%	60%	System efficiency is maximum as it uses a pre-trained Word2Vec model	Takes context into consideration by using word embeddings to represent semantic meaning

Figure 4: Ontology Algorithms Comparison

the concept itself. To address this gap, the algorithm needs to retain in memory the concept being extracted, instead of using naive pattern-matching approaches. Also, the analysis of the shortlisted research papers in each category state the declining research on Clustering and Set-Theoretic algorithms, and an increasing in research of learning algorithms.

5.1 Possible solution: Long Short Term Memory Networks

The algorithms proposed above use either pattern matching techniques, naive SVO (subject-verb-object) triplet extraction techniques or semantic similarity techniques for extracting concepts. All of these techniques were at the concept level, and though extension of algorithms like C-PANKOW used Doc2Vec to gauge similarity of documents, none of these algorithms involved understanding of the text corpus to filter out irrelevant data. Hence, we suggest a need to incorporate Deep Learning to enrich ontologies.

We propose a Deep Learning solution using Long Short Term Memory Networks (LSTMs) [5] to address the identified gaps. We explain our reason for proposing an LSTM with the help of an example.

"Cross-Frame Scripting (XFS) is a browser based attack that combines malicious JavaScript with an iframe while loading a legitimate site. This attack is one of the most common attacks against IE. This is due to it leaking keyboard events across HTML framesets.".

On passing these sentences to a concept-relationship extraction system, such as the ones described previously, a "one-of" relationship would be formed between "This attack" and "one of the most common attacks against IE" and a "due-to" relationship would be formed between "this" and "leaking keyboard events across HTML framesets". However, in the second sentence, "This attack" refers to "XFS attack" (from the first sentence) and is the concept identified and can be abstracted to "browser based attack". But in the third sentence, the current concept has changed and "this" refers to "attacks against IE". Hence, normal concept extraction techniques would not work for these examples, since the current concept may change every sentence. LSTMs can be trained to learn optimal forget matrices that continually update the cell state, thereby, enabling the model to maintain the state of a concept (by addding new concepts and removing old ones) for longer durations. Thus, LSTMs can enable greater semantic understanding as well as detection of long ranging patterns, which theoretically should improve precision.

6 Conclusion

We started this survey paper by describing the need for enrichment of ontologies. We proceeded to survey the existing domain literature in the field of ontology learning from text and got a subset of 166 research papers and 20 survey papers. From shortlisted 101 papers, we narrowed down to the 23 most relevant research papers. These 23 papers were classified into four categories based on the approach used for ontology enrichment, namely Clustering, Set-Theoretic, Web Corpus-based and Learning-based Algorithms. We selected a seminal paper from each category, based on criteria like the date of publication, the number of citations, relevance to our end goal etc. and then described the approach of the algorithms. Next,

[5] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

we compared algorithms performance (context relevance, precision and efficiency) on the enrichment of Information Security ontology. We found that with each trend, some of the gaps were overcome but there still remained the problem of retaining concepts to improve relevance in a scalable manner. We proposed LSTMs as a possible solution for concept retention, since they use a memory state to partially remember/forget concepts over long periods of time as require. In future, we plan on implementing the proposed LSTM model to improve precision and efficiency of the state-of-the-art. We also plan to validate further with complex ontologies, and extend our concept enrichment model to the addition of instances for building knowledge base.

References

Khalida Bensidi Ahmed, Adil Toumouh, and Mimoun Malki. 2012. Effective Ontology Learning: Concepts' Hierarchy Building using Plain Text Wikipedia. In *ICWIT*, pages 170–178.

Mazen Alobaidi, Khalid Mahmood Malik, and Maqbool Hussain. 2018. Automated Ontology Generation Framework Powered by Linked Biomedical Ontologies for Disease-Drug Domain. *Computer Methods and Programs in Biomedicine*.

Kamel Ben-Khalifa and Susanne Motameny. 2007. Horn-Representation of a Concept Lattice. volume 38.

Souad Benomrane, Zied Sellami, and Mounir Ben Ayed. 2016. An ontologist Feedback driven Ontology Evolution with an Adaptive Multi-agent System. *Advanced Engineering Informatics*, 30(3):337–353.

Meisam Booshehri and Peter Luksch. 2014. Towards Adding Linked Data to Ontology Learning Layers. In *Proceedings of the 16th International Conference on Information Integration and Web-based Applications & Services*, pages 401–409. ACM.

Meisam Booshehri and Peter Luksch. 2015. An Ontology Enrichment Approach by using DbPedia. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, page 5. ACM.

Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. 2005. Ontology Learning from Text: An overview. *Ontology Learning from Text: Methods, Evaluation and Applications*, 123:3–12.

Mercedes Argüello Casteleiro, Maria Jesus Fernandez Prieto, George Demetriou, Nava Maroto, Warren J Read, Diego Maseda-Fernandez, Jose Julio Des Diz, Goran Nenadic, John A Keane, and Robert Stevens. 2016. Ontology Learning with Deep Learning: a Case Study on Patient Safety Using PubMed. In *SWAT4LS*.

Jill Cheng, Melissa Cline, John Martin, David Finkelstein, Tarif Awad, David Kulp, and Michael A Siani-Rose. 2004. A Knowledge-based Clustering Algorithm Driven by Gene Ontology. *Journal of Biopharmaceutical statistics*, 14(3):687–700.

Philipp Cimiano, Siegfried Handschuh, and Steffen Staab. 2004. Towards the self-annotating web. In *Proceedings of the 13th international conference on World Wide Web*, pages 462–471. ACM.

Philipp Cimiano, Andreas Hotho, and Steffen Staab. 2005a. Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. *Journal of Artificial Intelligence Research*, 24:305–339.

Philipp Cimiano, Günter Ladwig, and Steffen Staab. 2005b. Gimme'the Context: Context-driven Automatic Semantic Annotation with C-PANKOW. In *Proceedings of the 14th international conference on World Wide Web*, pages 332–341. ACM.

Philipp Cimiano and Steffen Staab. 2005. Learning Concept Hierarchies from Text with a Guided Agglomerative clustering Algorithm. In *Proceedings of the Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods*.

Valdemar TF Confort, Kate Revoredo, Fernanda Araujo Baião, and Flávia Maria Santoro. 2015. Learning Ontology from Text: A Storytelling Exploratory Case Study. In *International Conference on Knowledge Management in Organizations*, pages 477–491. Springer.

Gaoying Cui, Qin Lu, Wenjie Li, and Yirong Chen. 2009. Mining Concepts from Wikipedia for Ontology Construction. In *Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 03, pages 287–290. IEEE.

Carmen De Maio, Giuseppe Fenza, Vincenzo Loia, and Sabrina Senatore. 2009. Towards an Automatic Fuzzy Ontology Generation. In *IEEE International Conference on Fuzzy Systems*, pages 1044–1049. IEEE.

Lucas Drumond and Rosario Girardi. 2010. An Experiment using Markov Logic Networks to Extract Ontology Concepts From Text. *lLearning*, 1:2.

Andreas Ekelhart, Stefan Fenz, Markus D Klemen, and Edgar R Weippl. 2006. Security Ontology: Simulating Threats to Corporate assets. In *International Conference on Information Systems Security*, pages 249–259. Springer.

Andreas Faatz and Ralf Steinmetz. 2002. Ontology Enrichment with Texts from the WWW. *Semantic Web Mining*, 20.

Bernhard Ganter and Rudolf Wille. 1996. Formal Concept Analysis. *Wissenschaftliche Zeitschrift-Technischen Universitat Dresden*, 45:8–13.

Bernhard Ganter and Rudolf Wille. 1999. Contextual Attribute Logic. In *International Conference on Conceptual Structures*, pages 377–388. Springer.

Hele-Mai Haav. 2004. A Semi-automatic Method to Ontology Design by Using FCA. In *CLA*. Citeseer.

Mohamed Rouane Hacene, Amedeo Napoli, Petko Valtchev, Yannick Toussaint, and Rokia Bendaoud. 2008. Ontology Learning from Text using Relational Concept Analysis. In *International MCETECH Conference on e-Technologies*, pages 154–163. IEEE.

Marti A Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th conference on Computational linguistics*, volume 2, pages 539–545. Association for Computational Linguistics.

Barbara Kitchenham. 2004. Procedures for Performing Systematic Reviews. *Keele, UK, Keele University*, 33(2004):1–26.

Jey Han Lau and Timothy Baldwin. 2016. An Empirical Evaluation of Doc2Vec with Practical Insights into Document Embedding Generation. *arXiv preprint arXiv:1607.05368*.

Chang-Shing Lee, Yuan-Fang Kao, Yau-Hwang Kuo, and Mei-Hui Wang. 2007. Automated Ontology Construction for Unstructured Text Documents. *Data & Knowledge Engineering*, 60(3):547–566.

Alexander Maedche and Valentin Zacharias. 2002. Clustering Ontology-based Metadata in the Semantic Web. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 348–360. Springer.

Kavi Mahesh, Sergei Nirenburg, et al. 1995. A Situated Ontology for Practical NLP. In *Proceedings of the IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*, volume 19, page 21. Citeseer.

Olena Medelyan, David Milne, Catherine Legg, and Ian H Witten. 2009. Mining Meaning from Wikipedia. *International Journal of Human-Computer Studies*, 67(9):716–754.

Hamid Mousavi, Deirdre Kerr, Markus Iseli, and Carlo Zaniolo. 2014. Harvesting Domain Specific Ontologies from Text. In *IEEE International Conference on Semantic Computing*, pages 211–218. IEEE.

Yael Netzer, David Gabay, Meni Adler, Yoav Goldberg, and Michael Elhadad. 2009. Ontology Evaluation through Text Classification. In *Advances in Web and Network Technologies, and Information Management*, pages 210–221. Springer.

Natalya F Noy and Michel Klein. 2004. Ontology Evolution: Not the same as Schema Evolution. *Knowledge and Information Systems*, 6(4):428–440.

Magnus Sahlgren. 2008. The Distributional Hypothesis. *Italian Journal of Disability Studies*, 20:33–53.

David Sánchez. 2010. A Methodology to Learn Ontological Attributes from the Web. *Data & Knowledge Engineering*, 69(6):573–597.

Zied Sellami, Valérie Camps, and Nathalie Aussenac-Gilles. 2013. DYNAMO-MAS: a Multi-agent System for Ontology Evolution from Text. *Journal on Data Semantics*, 2(2-3):145–161.

Ilaria Tiddi, Nesrine Ben Mustapha, Yves Vanrompay, and Marie-Aude Aufaure. 2012. Ontology Learning from Open Linked Data and Web Snippets. In *Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 434–443. Springer.

Gerhard Wohlgenannt and Filip Minic. 2016. Using word2vec to Build a Simple Ontology Learning System. In *International Semantic Web Conference (Posters & Demos)*.

Sanskrit Segmentation Revisited

Sriram Krishnan and Amba Kulkarni
Department of Sanskrit Studies
University of Hyderabad
sriramk8@gmail.com, apksh.uoh@nic.in

Abstract

Computationally analyzing Sanskrit texts requires proper segmentation in the initial stages. There have been various tools developed for Sanskrit text segmentation. Of these, Gérard Huet's Reader in the Sanskrit Heritage Engine analyzes the input text and segments it based on the word parameters - phases like iic, ifc, Pr, Subst, etc., and *sandhi* (or transition) that takes place at the end of a word with the initial part of the next word. And it enlists all the possible solutions differentiating them with the help of the phases. The phases and their analyses have their use in the domain of sentential parsers. In segmentation, though, they are not used beyond deciding whether the words formed with the phases are morphologically valid. This paper tries to modify the above segmenter by ignoring the phase details (except for a few cases), and also proposes a probability function to prioritize the list of solutions to bring up the most valid solutions at the top.

1 Introduction

Every Sanskrit sentence in the *saṃhitā* form (continuous sandhied text) is required to be segmented into proper morphologically acceptable words and the obtained result should agree with syntactic and semantic correctness for it's proper understanding. The obtained segmented text consists of individual words where even the compounds are segmented into their components. And there can be more than one segmentation for the same *saṃhitā* text. The segmented form does not provide any difference in the sense of the text when compared with the *saṃhitā* form except for the difference in the phonology of the words where it can be observed that the end part of the initial word together with the first letter of the next word undergoes phonetic change. The *saṃhitā* form, in fact, represents the text similar to a speech text because the knowledge transfer, in the olden days, was predominantly based on oral rendition. But now, for extracting information from these texts it is necessary that they be broken down into pieces so that the intention of the text is revealed completely without any ambiguity. In order to understand any Sanskrit text, this process of breaking down into individual words is necessary, and it is popularly known as *sandhi-viccheda* (splitting of the joint text) in Sanskrit.

This process takes into account the morphological analyses of each of the split-parts obtained. As there is always a possibility for multiple morphological analyses even for individual words, considering only the morphological validation might result in enormous number of solutions for long sentences. So, syntactical accuracy is also measured to reduce the number of solutions. Even then, there is always a possibility for multiple solutions to remain, which cannot be resolved further without the semantic and contextual understanding of the sentence (Hellwig, 2009). Owing to this, we find that there is non-determinism right at the start of linguistic analysis (Huet, 2009), since *sandhi* splitting is the first step in the analysis of a Sanskrit sentence.

This kind of non-determinism is also found in languages like Chinese and Japanese, where the word boundaries are not indicated, and also in agglutinative languages like Turkish (Mittal, 2010). In some of these languages

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 105–114
Hyderabad, India, December 2019.

like Thai (Haruechaiyasak et al., 2008), most of the sentences have mere concatenation of words. Possible boundaries are predicted using the syllable information, and the process of segmentation starts with segmenting the syllable first, followed by the actual word segmentation. For Chinese though, their characters called *hanzi* are easily identifiable, and the segmentation could be done by tagging (Xue, 2003), or determining the word-internal positions using machine learning or deep learning algorithms (like what is done in Ma et al. (2018)). In the case of Vietnamese (Thang et al., 2008), compound words are predominantly formed by semantic composition from 7000 syllables, which can also exist independently as separate words. This is similar to what can be observed in *aluk samāsa* in Sanskrit, which are rare in occurrence. For languages like English, French and Spanish where the boundaries are specifically observed as delimiters like space, comma, semi-colon, full stop, etc., segmentation is done using these delimiters and is comparatively simple.

In all the above cases, we find that either there are delimiters to separate the words, or individual words are joined by concatenation which ultimately rests the segmentation process in the identification of boundaries. In the case of Sanskrit though, these kinds of words form a very small percentage. Rather, there is the euphony transformation that takes place at every word boundary. This transition can be generally stated as u|v → w, where u is the final part of the first word, v the first part of the next word, and w the resultant form after combining u and v. Here the parts may contain at the most two phonemes. The resultant w may contain additional phonemes or may have elisions, but never are more than two phonemes introduced. So this transition or *sandhi* (external) occurs only at the phoneme level, and it does not require any other information regarding the individual words used.[1] But the reverse process of segmentation does require a morphological analyzer to validate the segments in a split. And it is entirely up to the speaker or writer to perform these transitions or keep the words separated (called *vivakṣā* - speaker's intention or desire). But in most of the texts and manuscripts, the *sandhi* is done throughout the text. So, finding the split location alone will not be enough to segment the texts properly.

Having looked into some of the intricacies of *sandhi* in Sanskrit, we can come up with a mechanical segmentation algorithm that splits a given text into all possible segments:

1. Traverse through the input text and mark all possible split locations which could be found in the list of sandhied letters.[2]

2. When a sandhied letter is marked, then list all it's possible splits.

3. Considering all the possible combinations of the words formed after each of these splits are allowed to join with the respective words (left word or right word), take each of the words, starting from the first word, to check for the morphological feasibility. Keep in mind that the words thus formed may also bypass the split locations, where they don't consider the split location present in between them.[3]

4. If the word is morphologically correct, then consider it as a valid split word and move on to the next split location, and do step 3 until the last word of the sentence is reached. The sequence of words thus formed is the first solution. If the word is not morphologically correct, move to step 5. If all the words formed in a single split location, either on the left or on the right, or both, are not morphologically correct, then discard that split location and move to step 3 for the next location.

[1] In the case of internal *sandhi* between preverbs and verbs, the lexical knowledge of the preverb is required. And in some compounds (like those denoting a *saṃjñā*), certain cases of retroflexion is permitted. But in this paper only the external *sandhi* is considered.

[2] To get the list of sandhied letters, there is a list of *sūtras* or rules for the joining of letters, available in Pāṇini's *Aṣṭādhyāyī* from which one can reverse analyze and obtain the list of sandhied letters.

[3] For example - *rāmālayaḥ* has split locations at 3 places - second, fourth (due to *akaḥ savarṇe dīrghaḥ in Aṣṭādhyāyī 6.1.101*) and sixth-seventh (due to *eco'yavāyāvaḥ in Aṣṭādhyāyī 6.1.78*) letters. So, *rā* is one split word, as also *rāma*, which bypasses the split location *ā*. Similarly, we can find other split words also.

5. Check the words formed from the subsequent splits and continue with steps 3 and 4 to obtain other solutions.

6. Trace back every split location, and perform step 5.

7. In this way, get all the possible combinations of the split words.

Although this mechanical process looks quite simple, the previously mentioned issues like non-determinism do prevail. And systems like the Sanskrit Reader in Sanskrit Heritage Engine come up with better ways to try to account these problems. The current paper tries to update these efforts. It is organized as follows: Section 2 gives the update on how the segmentation for Sanskrit has been dealt with in recent years. Section 3 discusses the important features of The Sanskrit Heritage Engine's Reader. Section 4 explains in detail the issues present in the Reader. The modifications needed to be done, and the implementation for this paper compose Section 5. It also quotes theoretically the reasons for these modifications and provides the probability function proposed in this paper. Section 6 describes the methodology of the implementation, and the results and observations are in Section 7.

2 Current Methods

Achieving the correct segmentation computationally is as much difficult as it is manually. A general approach would be the conversion of the mechanical *sandhi* splitting process mentioned in Section 1 to a working algorithm, followed by checking the statistics available for the frequencies of the words and transitions. But there has been a lot of better research work, both rule-based and statistical, on computational *sandhi* splitting in Sanskrit.

Huet (2003), as a part of the Sanskrit Heritage Engine, developed a Segmenter for Sanskrit texts using a Finite State Transducer. Two different segmenters were developed - one for internal *sandhi*, which is deployed in the morphological analyser, and the other for external *sandhi*. The current paper focuses on updating this external *sandhi* segmenter.

Mittal (2010) had used the Optimality Theory to derive a probabilistic method, and developed two methods to segment the input text
(1) by augmenting the finite state transducer developed using OpenFst (Allauzen et al., 2007), with *sandhi* rules where the FST is used for the analysis of the morphology and is traversed for the segmentation, and
(2) used optimality theory to validate all the possible segmentations.

Kumar et al. (2010) developed a compound processor where the segmentation for the compound words was done and used optimality theory with a different probabilistic method (discussed in section 5).

Natarajan and Charniak (2011) later modified the posterior probability function and also developed an algorithm based on Bayesian Word Segmentation methods with both unsupervised and supervised algorithms.

Krishna et al. (2016) proposed an approach combining the morphological features and word co-occurrence features from a manually tagged corpus from Hellwig (2009), and took the segmentation problem as a query expansion problem and used Path Constrained Random Walk framework for selecting the nodes of the graph built with possible solutions from the input.

Reddy et al. (2018) built a word segmenter that uses a deep sequence to sequence model with attention to predict the correct solution. This is the state of art segmenter with precision and recall as 90.77 and 90.3, respectively.

IBM Research team (Aralikatte et al., 2018), had built a Double Decoder RNN with attention as $seq2(seq)^2$, where they have emphasized finding the locations of the splits first, and then the finding of the split words. And they have the accuracy as 95% and 79.5% for finding the location of splits and the split sentence, respectively.

Hellwig and Nehrdich (2018) developed a segmenter using Character-level Recurrent and Convolutional Neural Networks, where they tokenize Sanskrit by jointly splitting compounds and resolving phonetic merges. The model does not require feature engineering or external linguistic resources. It works well with just the parallel versions of raw and segmented text.

Krishna et al. (2018) proposed a structured

prediction framework that jointly solves the word segmentation and morphological tagging tasks in Sanskrit by using an energy based model which uses approaches generally employed in graph based parsing techniques.

3 Heritage Segmenter

The Sanskrit Heritage Engine's Segmenter was chosen for further development, for three reasons -

1. It is the best segmenter available online with source code available under GPL.
2. It uses a Finite State Transducer, and hence the segmentation is obtained in linear time.
3. It can produce all possible segmentations that one can arrive at, following Pāṇini's rules for *sandhi*.

It analyses the given input and produces the split based on three main factors:

1. Morphological feasibility: whether each of the words observed as a split is morphologically obtainable.
2. Transition feasibility: whether every transition observed with each of the word is allowed.
3. Phase feasibility: whether the sequence of words have proper phase values. This is a constraint on the POS of a word. Although Sanskrit is a free word order language, there are certain syntactic constraints which govern the word formation, and the sequence of components within a word follows certain well defined syntax. The phase feasibility module takes care of this. Figure 1 shows a part of the lexical analyzer, developed by Goyal and Huet (2013), that portrays these phases like Iic, Inde, Noun, Root, etc.

Let us consider the sentence *rāmālayo′sti*, as an example to understand these factors. It can be observed that there are twelve possible split solutions given in Table 1, from which, all the observed split words are shown in Figure 2.

Other possible words like *rā, mālayaḥ*, etc. are not taken as proper splits because they do not form proper words according to the morphological analyzer present in the system. In this way, morphological feasibility is checked. In the same example, we find that, at the last possible split location represented by *o′*, we can split it as *aḥ* and *a* but not in any other way.[4] This is ensured by the transition feasibility module.

The phase details like iic for *rāma* or pr for *asti*, etc. are displayed along with the words. These assignments of the phase information to the words and their analysis are the jobs of the phase feasibility module. To understand these phases, look at Figure 3 (the first solution for the sentence *rāmālayo′sti*). *rāma* is the first split and has the phase iic. *ālayaḥ* is the second split with two morphological possibilities - *ālaya* and *āli*. And the transition between the first two words is - a | ā → ā.

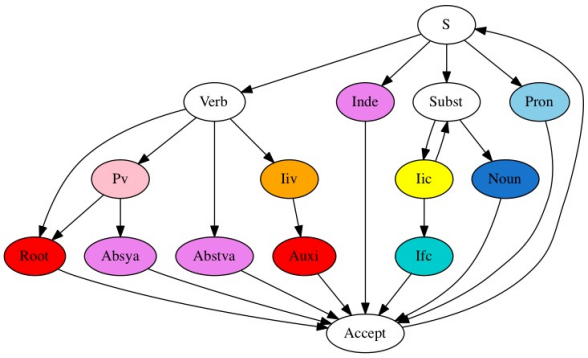


Figure 1: A simplified lexical analyzer

Solutions
rāma (iic) ālayaḥ (ālaya/āli masc) asti
rāma (iic) ālayaḥ (āli fem) asti
rāma (iic) alayaḥ (ali masc) asti
rāma (iic) a (iic) layaḥ asti
rāma (iic) alayaḥ (ali fem) asti
rāmā (fem) layaḥ asti
rāmā (fem) ālayaḥ (ālaya/āli masc) asti
rāmā (fem) alayaḥ (ali masc) asti
rāmā (fem) a layaḥ asti
rāma (rā) ālayaḥ (ālaya/āli masc) asti
rāma (rā) alayaḥ asti
rāma (rā) a (iic) layaḥ asti

Table 1: List of solutions for the sentence rāmālayo′sti

[4] For the rules governing these transitions refer the *Aṣṭādhyāyī sūtra: atororaplutādaplute (6.1.113) and haśi ca (6.1.114)*

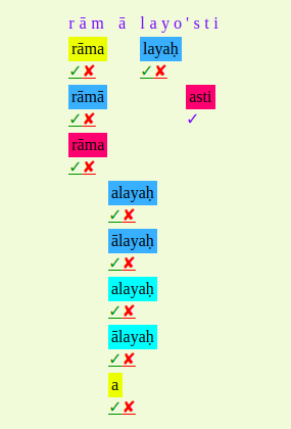


Figure 2: The interface for choosing or rejecting the obtained split words for the example *rāmālayo′sti*

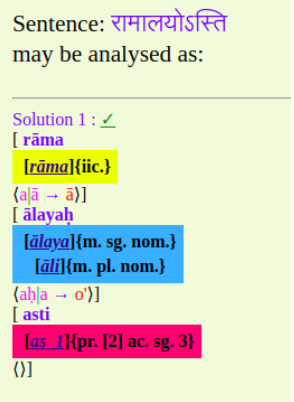


Figure 3: The first solution for the sentence rāmālayo′sti

The third split is *asti* with root *as* and phase pr. And the transition follows the equation - aḥ | a → o′. These transitions are taken care of by the transition feasibility module and the phases mentioned above are taken care of by the phase feasibility module.

According to Goyal and Huet (2013), sentences are formed by the image of the relation R (*sandhi* rules) on Kleene closure of W*, of a regular set W of words (vocabulary of inflected words). The Sanskrit Heritage Reader accepts a candidate sentence w, and applies the inverted form of the relation R, thus producing a set of words - w1, w2, w3,.... And each of the individual words are valid according to the rules of morphology, and their combination makes some sense.

The methodology followed in the Segmenter proposed in Goyal and Huet (2013) starts with using the finite state transducer for generating the chunks, instead of the traditional recursive method over the sentence employed in many *sandhi* splitting tools. The FST considers the phases as important characteristics of the words. These phases correspond to a finite set of forms.

To understand how a word is obtained, let us first take a small example of how the substantival forms (*subantas*) are obtained. A *subanta* is analysed as a nominal stem followed by a suffix. The nominal stem can be either an underived stem or a derived stem. In case of a derived stem, the derivation of this stem is also provided by the segmenter. A compound, for example, has a derived stem which contains a sequence of components followed by a nominal suffix. And three phases are present to represent the *subantas*:

1. Noun, that contains declined forms of autonomous atomic substantive and adjective stems, from the lexicon
2. Ifc, non-autonomous and used as right-hand component of a compound
3. Iic contains bare stems of nouns to be used as left component

This sequence of Subst → Noun → Accept, creates a noun word. And the sequence of Subst → Iic$^+$ → Ifc → Accept, creates a compound word. These sequences can be observed in Figure 1. In this way, forms from these phases are selected, and gluing them with *sandhi* rules, a word is obtained. Considering all such possible phases in Sanskrit, an automation transition graph is formed and is used to traverse through to find the possible split locations and words together.

4 Issues in Heritage Segmenter

The Segmenter is embedded in the Sanskrit Reader which displays all the outputs with the corresponding split word, it's phase and the transition involved with the subsequent word, except when the number of outputs is huge, in which case it shows only the summary. The Reader shows the distinction between words and phases based on verb, noun, iic, inde, etc,

but not between some of the case-markers. So, there is inconsistency in disambiguation: sometimes the phase is used for pruning out certain solutions, but in some cases, it is not. For example, *rāmovanaṅgacchati* produces the following 4 solutions:

Solution 1 :
[rāmaḥ [rāma]m. sg. nom. aḥ|v → ov]
[vanam [vana]n. sg. acc. | n. sg. nom. m|g → ṅg]
[gacchati [gam]pr. [1] ac. sg. 3]

Solution 2 :
[rāmaḥ [rāma]m. sg. nom. aḥ|v → ov]
[vanam [vana]n. sg. acc. | n. sg. nom. m|g → ṅg]
[gacchati [gacchat ppr. [1] ac. [gam]]n. sg. loc. | m. sg. loc.]

Solution 3 :
[rāmaḥ [rā_1]pr. [2] ac. pl. 1 aḥ|v → ov]
present [vanam [vana]n. sg. acc. | n. sg. nom. m|g → ṅg]
[gacchati [gam]pr. [1] ac. sg. 3]

Solution 4 :
[rāmaḥ [rā_1]pr. [2] ac. pl. 1 aḥ|v → ov]
[vanam [vana]n. sg. acc. | n. sg. nom. m|g → ṅg]
[gacchati [gacchat ppr. [1] ac. [gam]]n. sg. loc. | m. sg. loc.]

The segmenter provides segmentation and also does partial disambiguation. For example, *rāmaḥ* is ambiguous morphologically and the machine has correctly disambiguated the alternatives. We see the noun analysis of it in the first and second solutions, and the verbal analysis in third and fourth solutions. But we notice that the word *vanam* which is ambiguous between two morphological analyses, one with nominative case marker and the other with accusative marker, is not disambiguated. Goyal and Huet (2013) mention that the consideration of a word's similar declensions as different might result in more ambiguity, and the purpose of the segmentation is to find the morphologically apt words and hence they are taken as one.

If we look at these four solutions, at the word level, all of them correspond to *rāmaḥ vanam gacchati*. In order to decide the correct solution among the four, we need to do syntactico-semantic analyses that depend solely upon the linguistic or grammatical information in the sentence (Kulkarni, 2013).

5 Proposed Modification

We notice that the use of phase information results in multiple solutions. In order to choose the correct solution among them, one needs to look beyond the word analysis and look at the possible relations between the words. This is the domain of the sentential parser. Only a sentential parser can decide which of the segmentations with phase information is the correct one. Thus we do not see any advantage of having the phase information.

And in the interface, the system is not uniform in resolving the ambiguities. It uses certain morphologically different phases under a single word, like *vanam* in section 4. Additionally, in the options for selecting or rejecting the words, sometimes the depth of the graph goes so deep that, there is a chance to miss some solutions.

Here we would like to mention that some of the phase information is still relevant for segmentation. And this corresponds to the compounds. The phase information tells if something is a component of a compound or a standalone noun. There are a few phases such as iic, iif, etc. that we do not ignore. Barring these we ignore all other phases.

Therefore, we propose the following modifications in the segmenter:

1. Ignore the phase information that is irrelevant from segmentation point of view and merge the solutions that have the same word level segmentation.

2. Prioritize the solutions.

This is similar to the intention in Reddy et al. (2018) where the morphological and other linguistic details are not obtained, but the segmentation problem is seen as an end in itself.

This is also similar to what Huet (2009) did as an update for Gillon (2009) to the compound analyzer where Gillon (2009) uses the

dependency structure to get the tree form consisting of all the parts of the compound word. And Huet (2009) made the lexical analyzer to understand the compound as a right recursive linear structure of a sequence of components. This made sure that only the compound components are obtained, and not their relationship with each other. This helps in easier and faster segmentation, but the next level syntactic analysis cannot be done without the relationship information of the components. Similarly, the same approach has been extended to all words, and not just compound words, and the phase details are not considered as valid parameters to distinguish solutions. Such solutions were termed duplicates and hence removed.

Once the duplicates are removed, prioritization needs to be done. Many probabilistic measures have been proposed in the past to prioritize the solutions.

Mittal (2010) calculated the weight for a specific split s_j as

$$W_{sj} = \frac{(\prod_{i=1}^{m-1}(\hat{P}(c_i) + \hat{P}(c_{i+1})) \times \hat{P}(r_i))}{m} \quad (1)$$

where $\hat{P}(c_i)$ is the probability of the occurrence of the word c_i in the corpus. $\hat{P}(r_i)$ is the probability of the occurrence of the rule r_i in the corpus. And m is the number of individual components in the split s_j.

Kumar et al. (2010) uses the weight of the split s_j as

$$W_{sj} = \frac{(\prod_{i=1}^{m} \hat{P}(c_i)) \times (\prod_{i=1}^{m-1} \hat{P}(r_i))}{m} \quad (2)$$

Natarajan and Charniak (2011) proposed a posterior probability function, $\hat{P}(s)$, the probability of generating the split $s = \langle c_1 ... c_m \rangle$, with m splits, and rules $r = \langle r_1, ..., r_{m-1} \rangle$ applied on the input, where

$$\hat{P}(s) = \hat{P}(c_1) \times \hat{P}(c_2|c_1) \times \hat{P}(c_3|c_2, c_1) \times ... \quad (3)$$

$$\hat{P}(s) = \prod_{j=1}^{m} \hat{P}(c_j) \quad (4)$$

$\hat{P}(c_1)$ is the probability of occurrence of the word c_1. $\hat{P}(c_2|c_1)$ is the probability of occurrence of the word c_2 given the occurrence of the word c_1, and so on.

Mittal (2010) and Kumar et al. (2010) follow the GEN-CON-EVAL paradigm attributed to the Optimality Theory. This paper considers a similar approach but the probability function is taken as just the POP (product-of-products) of the word and transition probabilities of each of the solutions, discussed in section 6.

And to prioritize the solutions, the following statistical data was added from the SHMT Corpus:[5]

- *samāsa* words with frequencies
- *sandhi* words with frequencies
- *samāsa* transition types with frequencies
- *sandhi* transition types with frequencies

6 Methodology

Every solution obtained after segmentation is checked for the two details viz. the word and the transition (that occurs at the end of the word due to the presence of the next word), along with the phase detail that is checked only for those which correspond to the components of a compound. For every solution s, with output as

$$s = \langle w_1 . w_2 .... w_n \rangle$$

a confidence value, C_i, is obtained which is the product of the products of transition probablility (P_{ti}) and word probability (P_{wi}) for the word w_i,

$$C_i = \prod_{i=1}^{n} P_{wi} \times P_{ti} \quad (5)$$

The confidence value is obtained as follows:

- For every split word w_i, it's phase is checked to know whether the obtained word forms a compound or not.
- If it is a compound word, then it's corresponding frequency is obtained from compound words' statistical data, to calculate the word_probability, $P(w_i)$
- If it is not a compound word, then corresponding frequency is obtained from the *sandhi* words' statistical data.

[5] A corpus developed by the Sanskrit-Hindi Machine Translation (SHMT) Consortium under the funding from DeItY, Govt of India (2008-12). http://sanskrit.uohyd.ac.in/scl/GOLD_DATA/tagged_data.html

- For every transition associated with the word, the transition's corresponding frequency is obtained from either the *samāsa* transition data, or the *sandhi* transition data, based on the phase of the word; to calculate the transition_probability, $P(t_i)$.[6]

- The confidence value for the word, w_i is thus obtained as the product of word_probability and transition_probability $word\_probability \times transition\_probability$:

$$C_i = P_{w_i} \times P_{t_i} \tag{6}$$

- Finally the product of all such products was obtained for a single solution as the confidence value of the solution -

$$C_{total} = \prod_{i=1}^{n} P_{w_i} \times P_{t_i} \tag{7}$$

The solutions are then sorted as decreasing order of confidence values and the duplicates are removed based on only the word splits. The remaining solutions are displayed along with their number and confidence values.

7 Observations

The test data contained on the whole 21,127 short sandhied expressions, which were taken from various texts available at the SHMT corpus. This data was a parallel corpus of sandhied and unsandhied expressions. In case there are more than one segmentation possible, only one segmentation that was appropriate in the context where the sandhied expression was found is recorded.

The above data was fed to both the old and the modified segmenters. The results of the old segmenter were used as the baseline. A comparison was done on how the updated system performed with respect to the old system. The correct solution's position in the old segmenter was compared with the correct solution's position in the updated segmenter. Table 2 summarizes the results.

The old segmenter was able to correctly produce the segmented form in 19,494 cases out of the 21,127 instances. Of these, 53.51% of the solution was found to be in the first position, 12% in second position, and 9.61% in the third. All put together, 75.12% of the correct solutions were found in the top three solutions. Another important observation was that, the entire number of solutions taken all together was 2,40,942 for 21,127 test instances and the average number of solutions was 11.4 with the correct solution's position averaging at 4.71.

The modified segmenter was able to correctly produce the segmented form in 19,494 cases, same as the old segmenter. And 89.27% of the solution was found to be in the first position, 6.83% in the second position, and 2.2% in the third. All put together, 98.3% of the correct solutions were found in the top three solutions. This has an increase of 23.18% from the existing system. Also, the entire number of solutions taken all together was 1,46,610 for 21,127 test instances, having a drastic reduction of 94,332 solutions. The average number of solutions was 6.94, with the correct solution's position averaging at 1.18.

It can be noted that the overall Recall was 0.92270554267 for both the machines. Since only the statistics have been altered, the new system doesn't provide new solutions. Rather, it has increased the chances of getting the solution at the top three by 23.18%.

As we observe, the updated system reduces the total amount of solutions and brings up the most likely solutions. Also, we have more than 90% recall in both the cases. The missed out instances were either due to morphological unavailability or owing to the failure of the engine. Once the morphological analyzer is updated, there will definitely be a boost in the efficiency.

8 Conclusion

There are a few observations to be noted. First, by just using the POP (product of products) of the word and transition probabilities, we are able to obtain 98% precision. With better probabilities, we will definitely have better results. Second, this system can now be used to mechanically split the continuous texts like *Saṃhita-Pāṭha* of the Vedas or any other classical text to obtain the corresponding *Pada-Pāṭha*, which may be manually checked for

[6] If the frequency is not available for either the word or the transition, then it is assigned a default value of 1.

No. of	Old Segmenter	%	Updated Segmenter	%
Input text	21,127	-	21,127	-
Output text	21,127	-	21,127	-
Correct sol	19,494	92.27	19,494	92.27
Correct sol in 1st	10,432	53.51	17,403	89.27
Correct sol in 2nd	2,340	12.00	1,332	6.83
Correct sol in 3rd	1,874	9.61	429	2.20
Correct sol in 4th	937	4.8	164	0.84
Correct sol in 5th	703	3.6	73	0.37
Correct sol in sol > 5th	3,208	16.45	96	0.49
Incorrect sol	1,629	7.71	1,629	7.71
Entries with 1 solution	5,467	25.87	7,167	33.92
Entries with 2 solutions	3,320	15.71	4,002	18.94
Entries with 3 solutions	2,123	10.05	2,053	9.71

Table 2: A comparison of the performance of both the segmenters

correctness. Third, for mere segmentation, the phase distinctions were ignored, and the obtained solutions were prioritized. As stated earlier in the previous sections, to proceed to the next stage of parsing or disambiguation, we need more than just the split words. Thus this could be a proper base for working on how the available segmented words, along with the phase details, may be used for further stages of analysis.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. Openfst: A general and efficient weighted finite-state transducer library. In *Proceedings of International Conference on implementation and application of automata.*, Prague, Czech Republic.

Rahul Aralikatte, Neelamadhav Gantayat, Naveen Panwar, Anush Sankaran, and Senthil Mani. 2018. Sanskrit sandhi splitting using seq2(seq) 2. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.*

B. S. Gillon. 2009. Tagging classical sanskrit compounds. In *Sanskrit Computational Linguistics 3*, pages 98–105. Springer-Verlag LNAI 5406.

Pawan Goyal and Gérard Huet. 2013. Completeness analysis of a sanskrit reader. In *Proceedings, 5th International Sanskrit Computational Linguistics Symposium.*

Pawan Goyal and Gérard Huet. 2016. Design and analysis of a lean interface for Sanskrit corpus annotation. *Journal of Linguistic Modeling*, 4(2):117–126.

Pawan Goyal, Gérard Huet, Amba Kulkarni, Peter Scharf, and Ralph Bunker. 2012. A distributed platform for Sanskrit processing. In *24th International Conference on Computational Linguistics (COLING), Mumbai.*

Choochart Haruechaiyasak, Sarawoot Kongyoung, , and Matthew N. Dailey. 2008. A comparative study on thai word segmentation approaches. In *ECTI-CON.*

Oliver Hellwig. 2009. Sanskrittagger, a stochastic lexical and pos tagger for sanskrit. In *Lecture Notes in Artificial Intelligence.*

Oliver Hellwig and Sebastian Nehrdich. 2018. Sanskrit word segmentation using character-level recurrent and convolutional neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, page 2754–2763. Association for Computational Linguistics.

Gérard Huet. 2002. The Zen computational linguistics toolkit: Lexicon structures and morphology computations using a modular functional programming language. In *Tutorial, Language Engineering Conference LEC'2002.*

Gérard Huet. 2003. Lexicon-directed segmentation and tagging of sanskrit. In *XIIth World Sanskrit Conference, Helsinki, Finland. Final version in Themes and Tasks in Old and Middle Indo-Aryan Linguistics, Eds. Bertil Tikkanen and Heinrich Hettrich.*, pages 307–325, Delhi. Motilal Banarsidass.

Gérard Huet. 2005. A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *J. Functional Programming*, 15,4:573–614.

Gérard Huet. 2009. Sanskrit segmentation. In *South Asian Languages Analysis Roundtable*, Denton, Texas.

Malcolm D. Hyman. 2008. From pāṇinian sandhi to finite state calculus. In *Proceedings, 2nd International Sanskrit Computational Linguistics Symposium*, pages 253–265.

Amrith Krishna, Bishal Santra, Sasi Prasanth Bandaru, Gaurav Sahu, Vishnu Dutt Sharma4, Pavankumar Satuluri, and Pawan Goyal. 2018. Free as in free word order: An energy based model for word segmentation and morphological tagging in sanskrit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Amrith Krishna, Bishal Santra, Pavan Kumar Satuluri, Sasi Prasanth Bandaru, Bhumi Faldu, Yajuvendra Singh, and Pawan Goyal. 2016. Word segmentation in sanskrit using path constrained random walks. In *Proceedings of the ACL 2010 Student Research Workshop*.

Amba Kulkarni. 2013. A deterministic dependency parser with dynamic programming for Sanskrit. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 157–166, Prague, Czech Republic. Charles University in Prague Matfyzpress Prague Czech Republic.

Amba Kulkarni and K. V. Ramakrishnamacharyulu. 2013. Parsing Sanskrit texts: Some relation specific issues. In *Proceedings of the 5th International Sanskrit Computational Linguistics Symposium*. D. K. Printworld(P) Ltd.

Amba Kulkarni, Preethi Shukla, Pavankumar Satuluri, and Devanand Shukl. 2015. How free is free word order in sanskrit. In *The Sanskrit Library*.

Anil Kumar, Vipul Mittal, and Amba Kulkarni. 2010. Sanskrit compound processor. In *Proceedings, 4th International Sanskrit Computational Linguistics Symposium*.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art chinese word segmentation with bi-lstms. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908. Association for Computational Linguistics.

Prasenjit Majumder, Mandar Mitra, and Swapan K. Parui. 2006. Shallow syntax analysis in sanskrit guided by semantic nets constraints. In *Proceedings of International Workshop on Research Issues in Digital Libraries*, Kolkata. ACM Digital Library.

Vipul Mittal. 2010. Automatic sanskrit segmentizer using finite state transducers. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 85–90. Association for Computational Linguistics.

Abhiram Natarajan and Eugene Charniak. 2011. S3-statistical sam. dhi splitting. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 301–308. Association for Computational Linguistics.

Vikas Reddy, Amrith Krishna, Vishnu Dutt Sharma, Prateek Gupta, Vineeth M. R, and Pawan Goyal. 2018. Building a word segmenter for sanskrit overnight. In *CoRR*.

Peter Scharf, Anuja Ajotikar, Sampada Savardekar, and Pawan Goyal. 2015. Distinctive features of poetic syntax preliminary results. In *Sanskrit syntax, pages 305–324*.

Peter Scharf and Malcolm Hyman. 2009. *Linguistic Issues in Encoding Sanskrit*. Motilal Banarsidass, Delhi.

DINH Q. Thang, LE H. Phuong, NGUYEN T. M. Huyen, NGUYEN C. Tu, Mathias Rossignol, and VU X. Luong. 2008. Word segmentation of vietnamese texts: a comparison of approaches. In *LREC'08*, Marrakech, Morocco.

Jingkang Wang, Jianing Zhou, and Jie Zhou Gongshen Liu. 2019. Multiple character embeddings for chinese word segmentation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, pages 210–216*.

N Xue. 2003. Chinese word segmentation as character tagging. In *Computational Linguistics and Chinese Language Processing*, volume 8(1), pages 29–48.

Integrating Lexical Knowledge in Word Embeddings using Sprinkling and Retrofitting

Aakash Srinivasan[1]*, **Harshavardhan Kamarthi**[2]*, **Devi Ganesan**[2], **Sutanu Chakraborti**[2]
[1]Dept. of Computer Science, University of California, Los Angeles
[2]Dept. of Computer Science and Engineering, Indian Institute of Technology Madras
Email: {s.aakash3431, harshavardhan864.hk}@gmail.com, {gdevi, sutanuc}@cse.iitm.ac.in,

Abstract

Neural network based word embeddings, such as Word2Vec and GloVe, are purely data driven in that they capture the distributional information about words from the training corpus. Past works have attempted to improve these embeddings by incorporating semantic knowledge from lexical resources like WordNet. Some techniques like retrofitting modify word embeddings in the post-processing stage while some others use a joint learning approach by modifying the objective function of neural networks. In this paper, we discuss two novel approaches for incorporating semantic knowledge into word embeddings. In the first approach, we take advantage of Levy et al's work which showed that using SVD based methods on co-occurrence matrix provide similar performance to neural network based embeddings. We propose a sprinkling technique to add semantic relations to the co-occurrence matrix directly before factorization. In the second approach, WordNet similarity scores are used to improve the retrofitting method. We evaluate the proposed methods in both intrinsic and extrinsic tasks and observe significant improvements over the baselines in many of the datasets.

1 Introduction

Neural Network based models (Mikolov et al., 2013a; Pennington et al., 2014) have been hugely successful in generating useful vector representation for words which preserve their distributional properties in a given corpora. Improving the quality of word embeddings have led to better performance in many downstream language tasks. Considering the widespread uses of word embeddings, there have been a lot of interest in improving the quality of these embeddings by leveraging lexical knowledge such as synonymy, hypernymy, hyponymy, troponymy and paraphrase relations. This is accompanied by the availability of large scale lexical knowledge available in WordNet (Miller, 1995) and Paraphrase Database (PPDB) (Ganitkevitch et al., 2013).

In this paper, we propose two simple yet powerful approaches to incorporate lexical knowledge into the word embeddings. First, we propose a matrix factorization based approach which uses the idea of 'sprinkling' (Chakraborti et al., 2006, 2007) semantic knowledge into the word co-occurrence matrix. Second, we identify the weaknesses of the retrofitting model (Faruqui et al., 2014) and propose a few modifications that improves the performance. We demonstrate the strength of the proposed models by showing significant improvements in two commonly used intrinsic language tasks - word similarity and analogy, and two extrinsic tasks - named entity recognition (NER) and part of speech tagging (POS).

2 Related Works

Learning of word embeddings that capture distributional information has been vital to many NLP tasks. Prediction-based methods such as skip-gram (Mikolov et al., 2013a) and CBOW (Bengio et al., 2003) use neural language modelling for predicting a given word given its context words (or vice-versa) and extract the learned weight vectors as word embeddings. On the other hand, count-based methods derive a co-occurrence matrix of words in the corpus and use matrix factorization techniques like SVD to extract word representations (Levy and Goldberg, 2014). GloVe (Pennington et al., 2014) uses co-occurrence matrix to train word embeddings such that the dot product between any two words is proportional to the log probability of their co-occurrence.

The models that incorporate lexical knowledge

*Equal Contribution

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 115–123
Hyderabad, India, December 2019.

into the word embeddings can be broadly classified into two categories, namely post processing and joint learning. Post processing methods such as (Faruqui et al., 2014; Mrkšić et al., 2016) take the pre-trained word embeddings and modify them by injecting semantic knowledge. The retrofitting method (Faruqui et al., 2014) derives similarity constraints from WordNet and other resources to pull similar words closer together. Whereas, the counterfitting approach, (Mrkšić et al., 2016) also tries to push the antonymous words away from each other. These approaches consider only one-hop neighbours' relations. We improve upon this by considering multi-hop neighbours as well as use structural and information-based similarity scores to determine their relative importance in imposing similarty contraints to the word embeddings.

Joint learning approaches like (Yu and Dredze, 2014; Fried and Duh, 2014; Vashishth et al., 2018) learn word embeddings by jointly optimizing distributional and relational information. For instance, in Yu and Dredze (2014), the objective function consists of both the original skip-gram objective as well as prior knowledge from semantic resources to learn improved lexical semantic embeddings. The recent work by Vashishth et al. (2018) uses Graph Convolutional Networks (GCNs) to learn relations between words and outperforms the previous methods in many language tasks.

Sprinkling: Latent Semantic Indexing (LSI), also known as Latent Semantic Analysis (LSA), learns a distributional representation for words by performing Singular Value Decomposition (SVD) on the term-document matrix. However, the dimensions obtained from LSI are not optimal in a classification setting because it is agnostic to class label information of the training data. The sprinkling method introduced by Chakraborti et al., (2006) improves LSI by appending the class labels as extra features (terms) to the corresponding training documents. When LSI is carried out on this augmented term-document matrix, terms pertaining to the same class are pulled closer to each other. An extension of this method, called adaptive sprinkling (Chakraborti et al., 2007), allows to control the importance of specific class labels by appending them multiple times to the term-document matrix. For instance, in case of double sprinkling, we append the class labels twice to the matrix thus improving the weakly supervised constraints imposed by class labels.

3 Proposed Models

In this section, we discuss the proposed models to incorporate semantic knowledge into word embeddings.

3.1 SS-PPMI & DSS-PPMI

In this approach, we take advantage of Levy and Goldberg's work (2014) in which the authors have shown that the objective function used in Word2vec (Mikolov et al., 2013a) implicitly factorizes a Shifted PPMI (SPPMI) matrix. While there are many methods that attempt to inject semantic knowledge into neural word embeddings, to the best of our knowledge, we have not come across any work that tries to inject semantic knowledge into the SPPMI matrix. In its original form, the SPPMI matrix captures only distributional information. Hence, we are interested in analysing the impact of injecting semantic knowledge into the SPPMI matrix and the effectiveness of the resulting word embeddings.

Inspired from (Chakraborti et al., 2006, 2007), which exploits the class knowledge of the documents by 'sprinkling' label terms into the term-document matrix before matrix factorization, we modify the SPPMI matrix by adding reachability information from lexical knowledge bases such as WordNet and PPDB. In the lexical graphs obtained from these knowledge bases, words are connected by edges representing relations such as synonymy, hypernymy, etc. We say that a word v is reachable from another word u if and only if there exists a path between them in the lexical graph. More formally, let n be the size of the vocabulary. We define the reachability matrix $L_k \in \{0,1\}^{n \times n}$ to be a zero-one square matrix with each element $L_k(u,v)$ indicating if word v is reachable from word u within k hops in the lexical knowledge graph.

We concatenate the reachability matrix with the SPPMI matrix to obtain *Sprinkled Shifted - Positive PMI* (**SS-PPMI**). We then perform SVD on this augmented matrix to obtain the enriched word embeddings.

$$\textit{SPPMI} = max(PMI - \log(neg), 0) \quad (1)$$

$$\textit{SS-PPMI} = \textit{SPPMI} \circ L_k \quad (2)$$

$$\textit{SS-PPMI} \approx U_x \Sigma_x V_x^T \quad (3)$$

$$Embeddings = U_x \Sigma_x^p \quad (4)$$

where $\circ$ denotes the matrix concatenation operation, neg denotes the number of negative samples and x denotes the lower rank approximation of the *SS-PPMI* matrix. *SS-PPMI* matrix is of dimensions $n \times 2n$. Following the work of Levy et al., (2014), we have used p as 0.5 to obtain the word embeddings.

The original motivation for sprinkling technique (Chakraborti et al., 2006) was that documents of same class are brought closer by appending the class labels to term-document matrix. Likewise, words which have strong syntactic relations such as synonymy or antonymy have similar neighbourhood in graphs like WordNet. This translates to these word pairs having similar columns in the reachability matrix. Thus, appending reachability matrix to SPPMI matrix would bring such words closer.

We can further strengthen these constraint by adding the reachability matrix multiple times as done in adaptive sprinkling (Chakraborti et al., 2007). We performed experiments adding reachability matrix twice and we call the resulting matrix as *Doubly Sprinkled Shifted - Positive PMI* (**DSS-PPMI**), which will be of dimensions $n \times 3n$.

3.2 W-Retrofitting

Retrofitting was introduced by Faruqui et al., (2014) and is a method to add semantic information to pre-trained word vectors. The post-processing step modifies the word embeddings such that the embeddings of words with semantic relations between them are pulled towards each other. Formally, given the pre-trained vectors $\hat{Q} = (\hat{q_1}, \hat{q_2} \cdots \hat{q_n})$, and a knowledge base represented by the adjacency matrix A, we need to learn new vectors $Q = (q_1, q_2 \cdots q_n)$ such that following objective $\psi(Q)$ is minimized:

$$\psi(Q) = \sum_{i=1}^{i=n} (\alpha_i \|q_i - \hat{q_i}\|^2 + \sum_{j=1}^{j=n} \beta_{ij} A_{ij} \|q_i - q_j\|^2) \quad (5)$$

The objective is a convex function and we can find the solution using the efficient iterative update method used in Faruqui et al., (2014):

$$q_i = \frac{\sum_{j=1}^{j=n} A_{ij} \beta_{ij} q_j + \alpha_i q_i}{\sum_{j=1}^{j=n} A_{ij} \beta_{ij} + \alpha_i} \quad (6)$$

The β_{ij} term is usually assigned as $degree(i)^{-1}$. This choice of assigning weights can be done in a better way by learning from semantic knowledge sourcea such as WordNet.

We propose a modification to the retrofitting methods called **W-Retrofitting** (weighted retrofitting), where we use WordNet-based similarity scores to obtain a better setting of β_{ij}. For two words w_i and w_j with WordNet similarity score $Sim(i, j)$, β_{ij} is obtained by normalizing the similarity scores across neighbors and is given as: $\beta_{ij} = \frac{Sim(i,j)}{\sum_{j'} Sim(i,j')}$. Since a word can have multiple synsets, the similarity score is the maximum of the similarity scores of all possible pairs of synsets, taking one each from the two words. For information based similarity measures like Lin similarity we compute mutual information from a random subset of Wikipedia corpus containing 100,000 articles. Further, we extend our method to consider nodes which are atmost 2 hops from given node when computing weights.

Scores	Datasets
Similarity	RG65, WS353S, Simlex-999
Relatedness	WS353R, TR9856
No Distinguishing	MEN, RW, MTunk, WS353

Table 1: The characterization of scores given by different word similarity datasets

4 Experimental Setup

4.1 Intrinsic Evaluation

We evaluate the proposed models on word similarity and analogy tasks.

Word similarity: We use MEN (Bruni et al., 2014), MTunk (Radinsky et al., 2011), RG65 (Rubenstein and Goodenough, 1965), Rare Words(RW) (Luong et al., 2013), SimLex999 (Hill et al., 2015), TR9856 (Levy et al., 2015b), WS353 (Finkelstein et al., 2002), WS353S (Similarity), WS353R (Relatedness). Spearman correlation is used as evaluation metric.

Analogy: We evaluated analogy task with Google Analogy (Mikolov et al., 2013a), MSR Analogy (Mikolov et al., 2013b) and Semeval2012 datasets. We follow the standardized setup as explained in (Jastrzebski et al., 2017).

4.2 Sources of Knowledge

We used two sources of semantic knowledge: WordNet (Miller, 1995) and PPDB (Ganitkevitch et al., 2013). We used the same PPDB

knowledge source used in Faruqui et al., (2014). We used WordNet source knowledge from V. Batagelj (2004). The relations considered are synonymy, hypernymy, meronymy and verb entailment. PPDB has 84467 nodes and 169703 edges, WordNet source we used has 82313 nodes and 98678 edges.

We used the latest Wikipedia dump[1] containing 6 Billion wikipedia articles to generate the SPPMI matrix. We followed the same procedure as given in Levy et al., (2015a) and chose the number of negative samples to be default value of 5. In all of our experiments, we chose embedding dimension as 300, which is commonly used in the literature.

4.3 Baselines

We use the following baselines for comparison

1. **GloVe**: Our first baseline is the GloVe embeddings (Pennington et al., 2014) trained on the Wikipedia corpus retrieved from Stanford NLP group website[2].

2. **Retrofit**: We apply the retrofitting technique (Faruqui et al., 2014) on the GloVe embeddings where Wordnet or PPDB was as the source of word relations.

3. **SPPMI**: We perform SVD on the Shifted PPMI matrix (as mentioned in Section 3) without sprinkling.

4. **SynGCN** (Vashishth et al., 2018): This work uses Graph-convolution based methods to impart relational information between words and have shown state-of-art results in many benchmarks. We directly report the available results from the original paper which uses same evaluation benchmarks.

4.4 Extrinsic Evaluation

To further test the effectiveness of the different methods in grounding word meanings, we utilize the embeddings in following tasks. The neural network architectures used for each of the tasks are same as that used in Vashishth et al., (2018).

1. **Part-of-speech tagging (POS):** This task classifies each word of given sentence as one of the part-of-speech tags. We use the LSTM based neural architecture discussed in Reimers and Gurevych (2017) on the Penn treebank dataset (Marcus et al., 1994).

2. **Named-entity recognition (NER):** The goal of this task is to extract and classify named entities in the sentences as person, organisation, location or miscellaneous. We use the model proposed in Lee et al., (2018) on CoNLL-2003 dataset (Sang and Meulder, 2003).

5 Results and Analysis

5.1 SS-PPMI

Reachability Matrix is powerful in capturing semantic information: We proposed a simple sprinkling approach in which a zero-one matrix captures the k-hop reachability information between words in a lexical knowledge graph. In order to see how effectively the reachability matrix captures the lexical knowledge, we performed SVD on the reachability matrix and obtained the word embeddings. Table 2 shows the performance of the obtained embeddings on word similarity task, The dimension of embedding used is 300. Interestingly, we clearly observe that the embeddings obtained from the reachability matrix only (without SPPMI matrix) compete strongly with 300 dimensional pretrained GloVe embeddings on the similarity based datasets. The best performing model gives a Spearman correlation which is **0.19** more than GloVe in Simlex999. Similarly, in RG65 and WS353S, the reachability based embeddings compete well with GloVe. Between the choice of PPDB or WordNet as the lexical knowledge sources, PPDB seems to be more helpful. In general, the performance of reachability-based embeddings increases with increasing the number of hops on the similarity datasets.

In the case of relatedness datasets, the model competes poorly with the baseline-GloVe. This is quite expected as the reachability matrix doesn't capture any information about the word co-occurrence. These observations have been foundational to our proposed *SS-PPMI* and *DSS-PPMI* methods.

SS-PPMI and DSS-PPMI provide significant improvements in word similarity and analogy: Table 3 provides the results with *SS-PPMI* and *DSS-PPMI* approaches on word similarity task with embedding dimension as 300. We clearly observe that the proposed models defeat the baseline

[1] https://dumps.wikimedia.org/enwiki/latest/
[2] https://nlp.stanford.edu/projects/GloVe/

		Similarity			Relatedness		No Distinction			
Lexical Knowledge	Hops - k	SimLex999	WS353S	RG65	WS353R	TR9856	WS353	MEN	MTurk	RW
Baseline - GloVe	-	0.370	**0.665**	**0.769**	**0.560**	**0.575**	**0.601**	**0.737**	**0.633**	**0.411**
PPDB	1	0.507	0.461	0.433	0.127	0.273	0.336	0.284	0.181	0.465
	2	**0.529**	0.567	0.512	0.128	0.261	0.362	0.304	0.261	0.506
WordNet	1	0.077	0.343	0.110	0.151	0.128	0.293	0.161	0.063	0.056
	2	0.209	0.349	0.378	0.163	0.149	0.285	0.275	0.145	0.209

Table 2: Performance of the reachability-based embeddings on similarity datasets. Reported numbers are the Spearman correlation coefficients.

			Similarity			Relatedness		No Distinction			
Method	Lexical Knowledge	hops	SimLex999	WS353S	RG65	WS353R	TR9856	WS353	MEN	MTurk	RW
SPPMI	-	-	0.385	0.728	0.783	0.603	0.625	0.663	0.742	0.599	0.516
SynGCN	-	-	0.455	**0.732**	-	0.457	-	0.601	-	-	0.337
SS-PPMI	PPDB	1	0.386	0.728	0.782	0.604	0.625	0.663	0.742	0.599	0.516
		2	0.398	0.733	0.775	0.619	0.628	0.669	0.743	0.610	**0.521**
DSS-PPMI	PPDB	1	0.386	0.728	0.782	0.604	0.625	0.663	0.742	0.599	0.516
		2	**0.420**	0.733	0.780	0.620	**0.629**	0.668	0.743	0.607	0.528
SS-PPMI	WordNet	1	0.393	0.724	0.792	0.627	0.597	0.667	0.769	0.611	0.464
		2	0.394	0.733	0.793	0.629	0.601	0.671	0.770	0.616	0.435
DSS-PPMI	WordNet	1	0.393	0.724	0.792	0.627	0.597	0.667	0.769	0.611	0.463
		2	0.394	**0.739**	**0.804**	**0.638**	0.599	**0.677**	**0.771**	**0.619**	0.414

Table 3: Results on word similarity datasets using SS-PPMI and DSS-PPMI embeddings

in all the datasets. The margin of improvement is quite high in case of similarity datasets. We see close to **0.21** increase in spearman correlation for Simlex999, **0.04** increase in RG65. This is somewhat expected as we already saw that reachability matrix contains lexical information. Interestingly, we also saw improvements in relatedness datasets where the sprinkling approaches perform narrowly better than SPPMI based approach. In other datasets like WS353, MEN we see improvements of about 0.02 and 0.03 in spearman correlation respectively. Overall, sprinkling significantly improves the performance on word similarity task.

Overall, we observe that Double Sprinkling method (*DSS-PPMI*) works better than *SPPMI* in word similarity task. Increasing the number of hops (k) in the reachability matrix improves the performance in word similarity , in general.

Table 4 shows improvements provided by the sprinkling methods on analogy datasets. We observe marginal improvements over baseline in google and SemEval2012.

5.2 W-Retrofitting

We apply our W-retrofitting model to GloVe (Pennington et al., 2014) embeddings trained on Wikipedia corpus. We experimented with one hop and two hop neighbors and several methods for similarity estimation: inverse path similarity, Jaing-Conrath Similarity (Jiang and Conrath, 1997), Wu -Palmer Similarity (Wu and Palmer, 1994), Leacock-Chowdorov Similarity (Leacock and Chodorow, 1998) and Lin Similarity (Lin et al., 1998). The neighbourhood information for estimating similarity was obtained from either WordNet or PPDB graphs. *We found that Jaing-Conrath Similarity works best for WordNet, inverse path similarity for PPDB*. So, we report results for these similarity measures only.

Method	Graph	hops	Google	SemEval
SPPMI-Baseline	-	-	0.337	0.176
SynGCN			-	**0.234**
SS-PPMI	PPDB	1	0.338	0.175
		2	**0.347**	**0.180**
DSS-PPMI	PPDB	1	0.338	0.176
		2	0.343	0.188
SS-PPMI	WordNet	1	0.122	0.166
		2	0.121	0.165
DSS-PPMI	WordNet	1	0.122	0.166
		2	0.118	0.161

Table 4: Analogy results using proposed SS-PPMI and DSS-PPMI approaches

Word Similarity: The performances of all our models are either comparable or superior to baselines as seen in table 5. We see that using PPDB knowledge source and path based similarity as weights in the retrofit objective functions gives the best performance and outperforms the baselines in most benchmarks.

Analogy: Some of our models outperform retrofitting baselines in Google analogy. In SemEval task, we mostly outperform GloVe but

		Similarity				Relatedness			No Distinction		
Method	Lexical Knowledge	Hops	SimLex999	WS353S	RG65	WS353R	TR9856	MTurk	WS353	MEN	RW
GloVe-baseline		-	0.37	0.665	0.769	0.56	0.575	0.633	0.601	0.737	0.411
SynGCN	-	-	0.455	**0.732**	-	0.457	-	0.601	-	-	0.337
Retrofit-baseline	PPDB	1	0.496	0.7	**0.825**	**0.585**	**0.601**	**0.675**	0.631	0.764	**0.431**
W-retrofit(path)		1	**0.509**	**0.71**	0.824	0.583	0.584	0.669	**0.641**	**0.773**	0.417
		2	0.422	0.628	0.788	0.519	0.525	0.63	0.562	0.722	0.372
Retrofit-baseline	Wordnet	1	**0.434**	0.693	0.774	**0.557**	0.574	**0.642**	0.607	**0.766**	0.387
W-retrofit(jcn)		1	0.432	0.685	0.772	0.543	0.568	0.64	0.6	0.764	0.353
		2	0.399	**0.73**	**0.785**	0.528	**0.579**	0.634	**0.616**	0.764	**0.389**

Table 5: Word Similarity results for W-Retrofitting approach

retrofitting baseline on WordNet gives the best score. The results are summarised in table 6

Similarity	Graph	Hops	Google	SemEval
GloVe		0	**0.717**	0.164
SynGCN			-	**0.234**
Retrofit-baseline	PPDB	1	0.451	0.171
path		1	0.448	0.167
		2	0.248	0.151
Retrofit-baseline	WordNet	1	0.603	**0.184**
jcn		1	**0.701**	0.161
		2	0.693	0.155

Table 6: Analogy results for W-Retrofitting

Model	SimLex999	WS353S	RG65
SPPMI	0.276	0.624	0.671
Retrofitting	0.336	0.624	0.752
W-Retrofitting	0.429	0.656	0.747
Reachability Matrix	0.561	0.567	0.664
Sprinkling	**0.591**	**0.748**	**0.821**
Model	WS353R	TR9856	MTurk
SPPMI	0.509	0.527	0.626
Retrofitting	0.479	0.534	0.623
W-Retrofitting	0.521	0.548	**0.631**
Reachability Matrix	0.194	0.325	0.283
Sprinkling	**0.638**	**0.629**	0.619
Model	WS353	MEN	RW
SPPMI	0.562	0.691	0.359
Retrofitting	0.545	0.708	0.350
W-Retrofitting	0.595	0.726	0.384
Reachability Matrix	0.376	0.325	0.506
Sprinkling	**0.682**	**0.771**	**0.560**

Table 7: Comparison with various baselines for word similarity and relatedness.

5.3 Overall Comparison on Word Similarity

In order to make fair and direct comparison between Sprinkling and Retrofitting, we applied retrofitting and W-retrofitting (using inverse-path similarity over PPDB graph) on the 300 dimensional SPPMI vectors. Table 7 provides the best results of the models on each of the word similarity and analogy datasets. We make the following observations. W-Retrofitting does much better than Retrofitting in similarity datasets, as what we saw with GloVe embeddings. The source of the improvement comes comes from two things: inclusion of two-hop neighbor information and the intelligent choice of weights from WordNet in W-Retrofitting.

Using only the Reachability Matrix provides very good scores in similarity based datasets, but doesn't capture relatedness information at all. Using sprinkling approach, we manage to obtain embeddings that have optimal combination of similarity and relatedness information and this makes it perform better than all the other baselines in similarity, relatedness and analogy tasks.

Method	Graph	Hops	NER	POS
SPPMI-Baseline			82.3	92.9
SS-PPMI	PPDB	1	83.4	93.3
		2	**84.7**	93.4
DSS-PPMI	PPDB	1	82.3	**93.5**
		2	87.3	93.4
SS-PPMI	Wordnet	1	83.5	92.8
		2	83.9	93.2
DSS-PPMI	Wordnet	1	83.2	93.2
		2	83.5	93.1

Table 8: Results on Extrinsic Evaluation tasks using SS-PPMI and DSS-PPMI embeddings

Method	Graph	Hops	NER	POS
GloVe	-		89.1	94.6
SynGCN	-		**89.5**	**95.4**
Retrofit-baseline	PPDB	1	88.8	94.8
path		1	88.7	95
		2	89.2	95.1
Retrofit-baseline	Wordnet	1	88.2	94.5
jcn		1	88.9	95
		2	**89.4**	**95.3**

Table 9: Results on Extrinsic Evaluation tasks using W-Retrofitting

5.4 Evaluation on Extrinsic tasks

The results on extrinsic tasks (discussed in Section 4.4) are given in Tables 8 and 9. In the case of sprinkling methods, we see that there is a clear in-

crease in scores for both the extrinsic tasks from using the proposed SS-PPMI matrix over using only the SPPMI matrix. We also see that models using PPDB perform better. One reason why we do not compare scores of sprinkling based methods with that of GloVe and Retrofitting based ones is that the vocabulary size(number of nodes) in PPDB or Wordnet graphs are lower than that for GloVe. We also didn't consider punctuation symbols in SPPMI unlike GloVe.

In the case of W-retrofitting, scores from the proposed W-Retrofitting model using jcn weights on wordnet graph are very similar to SynGCN model inspite of SynGCN being a more complex model with a lot of hyperparameters. We also see that the other methods of W-retrofitting have comparable performance to SynGCN. We observe improved performance by considering upto 2 hop neighbours over methods considering just 1 hop neighbours. It is quite interesting to see that the proposed light-weight retrofitting model competes strongly with the more complex SynGCN method as shown by the results in Table 9.

6 Conclusion and Future Work

In this paper, we proposed two simple yet powerful approaches to incorporate lexical knowledge into word embeddings. The first approach is a matrix factorization method that 'sprinkles' higher order graph information into the word co-occurrence and we show that it significantly improves the quality of the word embeddings. Second, we proposed a simple modification to the retrofitting method that improves it performance visibly. We showed the improvements of the proposed models over baselines in a variety of word similarity and analogy tasks, and across two popular lexical knowledge bases.

For extrinsic tasks, W-retrofitting showed comparable performance to the state-of-art SynGCN model, (Vashishth et al., 2018) inspite of SynGCN being a more sophisticated model with lots of parameters that constitute the weights of Graph Convolutional layers and linear layers of neural network used as well as many hyperparameters needed for training the neural network (such as number of GCN layers and their dimensions, learning rate, number of epochs, etc.).

In our sprinkling approach, we didn't consider any importance weighting for different relations. One promising direction that can be experimented in future is to use wordnet similarity scores or a combination of co-occurrence and lexical information as importance values in the reachability matrix. We could also use 'adaptive sprinkling' (Chakraborti et al., 2007) to give more importance to relations of specific sets of words.

The more recent methods that achieve the state-of-art results in a variety of language tasks utilize pre-trained models such as Elmo (Peters et al., 2018), BERT (Devlin et al., 2018) and XLNet (Yang et al., 2019). These models that learn context dependent word embeddings are pre-trained for different language tasks and are later fine-tuned for specific tasks. Another direction of research we would like to explore is to study the improvements gained by using our proposed models to initialize the word embeddings before pre-training these models.

References

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

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Sutanu Chakraborti, Robert Lothian, Nirmalie Wiratunga, and Stuart Watt. 2006. Sprinkling: supervised latent semantic indexing. In *European Conference on Information Retrieval*, pages 510–514. Springer.

Sutanu Chakraborti, Rahman Mukras, Robert Lothian, Nirmalie Wiratunga, Stuart NK Watt, and David J Harper. 2007. Supervised latent semantic indexing using adaptive sprinkling. In *IJCAI*, volume 7, pages 1582–1587.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131.

Daniel Fried and Kevin Duh. 2014. Incorporating both distributional and relational semantics in word representations. *arXiv preprint arXiv:1412.4369*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Stanisław Jastrzebski, Damian Leśniak, and Wojciech Marian Czarnecki. 2017. How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks. *arXiv preprint arXiv:1702.02170*.

Jay J Jiang and David W Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283.

Kenton Lee, Luheng He, and Luke S. Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *NAACL-HLT*.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015a. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Ran Levy, Liat Ein-Dor, Shay Hummel, Ruty Rinott, and Noam Slonim. 2015b. Tr9856: A multi-word term relatedness benchmark. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 419–424.

Dekang Lin et al. 1998. An information-theoretic definition of similarity. In *Icml*, volume 98, pages 296–304. Citeseer.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Mitchell P. Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *HLT*.

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, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

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.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *EMNLP*.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *ArXiv*, cs.CL/0306050.

A. Mrvar V. Batagelj. 2004. Wordnet transformed in pajek format, http://vlado.fmf.uni-lj.si/pub/networks/data./dic/wordnet/wordnet.htm, accessed on 10 april, 2019.

Shikhar Vashishth, Manik Bhandari, Prateek Yadav, Piyush Rai, Chiranjib Bhattacharyya, and Partha Pratim Talukdar. 2018. Incorporating syntactic and semantic information in word embeddings using graph convolutional networks. In *ACL*.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *ArXiv*, abs/1906.08237.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 545–550.

Robust Deep Learning Based Sentiment Classification of Code-Mixed Text

Siddhartha Mukherjee, Vinuthkumar Prasan, Anish Nediyanchath, Manan Shah, Nikhil Kumar
Samsung R&D Institute India, Bangalore
{siddhartha.m, vinuth, anish.n, mp.shah, nik.kumar} @samsung.com

Abstract

India is one of unique countries in the world that has the legacy of diversity of languages. English influence most of these languages. This causes a large presence of code-mixed text in social media. Enormous presence of this code-mixed text provides an important research area for Natural Language Processing (NLP). This paper proposes a novel **A**ttention based deep learning technique for Sentiment **C**lassification on **C**ode-**M**ixed **T**ext (ACCMT) of Hindi-English. The proposed architecture uses fusion of character and word features. Non-availability of suitable word embedding to represent these Code-Mixed texts is another important hurdle for this league of NLP tasks. This paper also proposes a novel technique for preparing word embedding of Code-Mixed text. This embedding is prepared with two separately trained word embeddings on romanized Hindi and English respectively. This embedding is further used in the proposed deep learning based architecture for robust classification. The Proposed technique achieves 71.97% accuracy, which exceeds the baseline accuracy.

1 Introduction

Languages used in India belong to several language families. Historical presence of British on Indian soil has led to a very high influence of English language on many of these Indian languages. People belonging in a multi-lingual society of India, gives rise of a large amount of text in various social media (Patra, 2018). Inclusion of English is very common in these texts. Essentially, an utterance in which a user makes use of grammar, lexicon or other linguistic units of more than one language is said to have undergone code-mixing (Chanda, 2016). Hindi is the widely spoken language of India and used in various media. The number of native Hindi speakers is about 25% of the total Indian population; however, including dialects of Hindi termed as Hindi languages, the total is around 44% of Indians, mostly accounted from the states falling under the Hindi belt[1]. This community contributes a large amount of text on social media. The form of Hindi language used in Social Media is mixed with English and are available in roman scripts. According to the study (Dey, 2014) most common reason for this kind of code mixing in a single text is 'Ease of Use'. The code-mixed Hindi and English language poses various types of challenges (Barman, 2014), which makes the text classification task on code-mixed text, an exciting problem in NLP Community. Despite a wide research on classification of code mixed texts, there remains open opportunities with two major aspects; first technique of preparing word embedding on Code-Mixed texts and second utilization of character and word features together to improve the accuracy. This research targets these two open points for exploration.

2 Related Work

Various research works have tried to tackle these challenges. Recent work of Prabhu (2016) utilizes character level LSTMs to learn sub word level information of social media text. Then this information is used to classify the sentences using an annotated corpus. The work is very interesting and achieves good accuracy. However the work does not intend to capture the information related to word level semantics. This provides a further scope of research to study the impact of word

[1]https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers_in_India

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 124–129
Hyderabad, India, December 2019.

embedding based approach on classification of code-mixed text. Sharma (2015) used an approach of lexicon lookup for text normalization and sentiment analysis on Code-Mixed text. Pravalika (2017) used lexicon lookup approach for domain specific sentiment analysis. These lexicon lookup based approaches lack capability to handle misspelled words and wide variety of these code mixed texts. Recent work (Lal, 2019) have used BiLSTM based dual encoder networks to represent the character based input and additional feature network to achieve good accuracy on code-mixed texts. Recent work (Yenigalla, 2018) has explored the opportunity of using both character and word embedding based feature to handle unknown words for text classification on monolingual English only text corpora. However, this approach is not common for Code-Mixed text, primarily because of the non-availability of word embedding for the Code-Mixed texts.

3 Dataset

We have considered Hi-En Code-Mixed dataset[2], shared by Prabhu (2016) as a baseline for this research.

3.1 Description

The dataset was collected from public Facebook pages of famous Indian personalities i.e. Salman Khan and Narendra Modi. The data is present in Roman script. The dataset contains 3879 comments. Each data is annotated with a 3-level of polarity scale i.e. Positive, Neutral and Negative. The dataset contains 15% negative, 50% neutral and 35% positive. Table 1 shows some example of code-mixed texts dataset.

Example	Approx. meaning in English	Polarity
Sir yeh tho sirf aap hi kar sakte hai. Great sir	Sir only you can do it. Great Sir	Positive
Kuch nahi karoge tum india ke liye	You won't do anything for India	Negative
Humari sabhayata humari pehchaan ...	Our civilization is our identity	Neutral

Table 1: Example from Hi-En Code-Mixed dataset.

3.2 Challenges

Transliteration of phonetic languages, like Hindi, into roman script creates several variations of the same word. For example, "बहुत" in Hindi which means "more" in English can be transliterated as "bahut", "bohoot" or "bohut" etc.

The Romanized Code-Mixed text, available on social media imposes additional challenges of contraction of phrases. For example, 'awsm' is shortened form of 'awesome'; 'a6a' is contracted from 'accha' etc. Romanized code-mixed text also contain sentences with non-grammatical constructs like 'Bhai jaan bolu naa.. yar' as well as non-standard spelling such as 'youuuu', 'jaaaaan' etc.

The phonetic similarity of various words across participant languages in the Code-Mixed text increases the challenge by introducing disambiguation for meaning of a word. For example, "man" in English means 'an adult human male' where as in Hindi it means 'mind'.

Large availability of clean corpora has given a rise in various kinds of research for Mono-lingual texts like English. On the other hand, the limited availability of clean & standard Code-Mixed corpus restricts wide spectrum of experiments, which depends on word-embedding based input.

3.3 Character Set

The dataset is cleaned of any special characters for this research. Final character set is of 36 characters including 26 English letters and 10 numbers. Final character set is:
abcdefghijklmnopqrstuvwxyz0123456789

4 Proposed Method

The proposed method consists of two major parts. First one is preparing a suitable word-embedding of code-mixed text and later one is a robust deep learning architecture for classification on code-mixed text.

4.1 Word-Embedding

There are three main aspects for preparing word embedding for Hindi-English Code-Mixed Texts. First is preparation of a corpus of Hindi Romanized text. Second one is preparing word embedding by choosing a right algorithm of word embedding.

[2] https://github.com/DrImpossible/Sub-word-LSTM

Third, is to ensure that words from both participant languages which are similar has nearby representation. To address the first aspect, we use Indic transliteration[3] on large Hindi-English corpus[4] where the Hindi text is present in Devanagari[5] script also contains English content. In this way, we achieve the Hindi-English Code-Mixed corpus in Roman Scripts. Figure 1 depicts the process of generating the desired corpus.

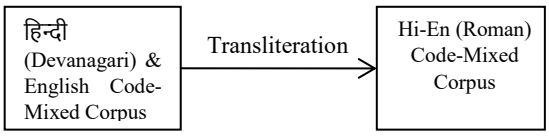


Figure 1: Corpus Preparation for Hi-En Code-mixed Text in Roman Script.

We hypothesize that the transliterated corpus represents a new language of Romanized Hindi. As discussed earlier there are various challenges of Romanized representation of Code-Mixed text such as presence multiple homo-phonic representations of a single word etc., so we have chosen fastText (Bojanowski, 2017) word representation as best method to train word embedding. This addresses the second aspect of previously discussed task of preparing word embedding. Once the corpus is generated, we have trained word embedding with fastText[6]. This trained embedding is capable of providing the vectorized representation of a Romanized Hindi word. On the other side, an utterance in the Code-Mixed corpus also contains English words as well. For example, the 1st utterance in the Table 1 contains two phrases, where 1st phrase contains the Romanized Hindi words and the 2nd phrase contains English words. This is the third and final aspect, discussed as a part of task of word embedding. Now to represent such an utterance using word embedding, we need the bi-lingual word embedding which include Romanized Hindi and English words as well. To cater to this requirement, we have used the proposed method (Smith, 2017) to represent bi-lingual representation of word from two monolingual representations. SVD is used to learn a linear transformation (a matrix), which aligns monolingual vectors from two languages in a single vector space[7]. In this experiment, we considered two monolingual word embedding(s). First is the trained word embedding of Romanized Hindi. Second one is the pre-trained & published[8] English word-embedding (Mikolov, 2018), which is trained on Wikipedia corpus.

4.2 Model Architecture

We prepare **A**ttention based deep learning architecture for **C**lassification of **C**ode-**M**ixed **T**ext (ACCMT) which uses learning from both character and word based representation. The proposed architecture consists of two major parts. The first part learns the sub-word level features from input character sequences. The other parts uses prepared word embedding as input and learn the word level features.

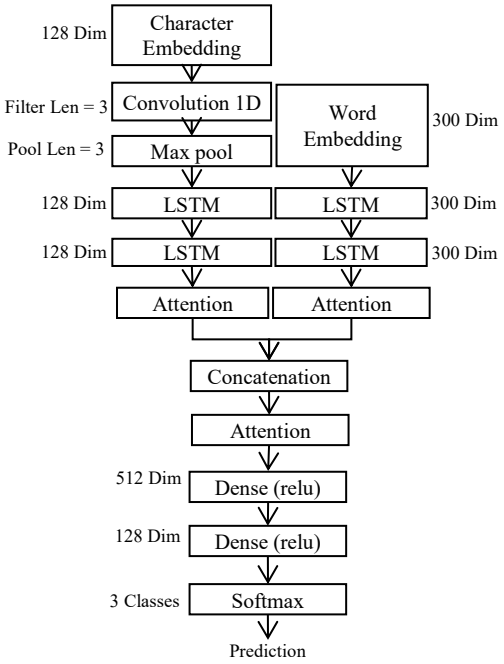


Figure 2: **A**ttention based deep learning architecture for **C**lassification of **C**ode-**M**ixed **T**ext (ACCMT)

The first part is similar as the baseline implementation Prabhu (2016), which is inspired by research work of Kim (2016). This part is independent of word vocabulary, which helps to resolve important issues in code mixed text like non-standard spelling, phrasal contraction etc.

[3] https://github.com/sanskrit-coders/indic_transliteration
[4] https://www.kaggle.com/pk13055/code-mixed-hindienglish-dataset
[5] https://en.wikipedia.org/wiki/Devanagari
[6] https://fasttext.cc/docs/en/python-module.html
[7] https://github.com/Babylonpartners/fastText_multilingual
[8] https://fasttext.cc/docs/en/pretrained-vectors.html

Even though this representation lack word level semantic interpretability, the assumption is that character n-gram serve semantic functions e.g. 'cat+s=cats'.

Formally a Sentence S is made of sequence of characters $[c_1, \dots, c_l]$where l is sentence length. $Q \in \mathbb{R}^{d \times l}$ is the representation of sentence where d being the dimension of character embedding. We perform the convolution of Q with filter $H \in \mathbb{R}^{d \times m}$ of length m. This operation provides a feature map $f \in \mathbb{R}^{l-m+1}$. Convolution is shown with '$*$' Operator in equation 1.

$$f = Q * H \tag{1}$$

Next max-pool operation of p features from f brings sub-word representation y.

$$\begin{aligned} a^t &= \tau_0 \times \tanh\left(\tau_u \widetilde{C_t} + \tau_f \widetilde{C_{t-1}}\right) \\ Where, \widetilde{C_t} &= \tanh(W_c[a^{t-1}, y^t] + b_c) \\ \tau_o &= \sigma(W_o[a^{t-1}, y^t] + b_o) \\ \tau_u &= \sigma(W_u[a^{t-1}, y^t] + b_u) \\ \tau_f &= \sigma(W_f[a^{t-1}, y^t] + b_f) \end{aligned} \tag{2}$$

Here y^t represents the input at current timestamp. Output from LSTM is a^tat time t. τ_o, τ_u, τ_f are respectively the output, input and forget gates of LSTM cell. $\widetilde{C_t}$ is the cell state at time t.

The second part is designed with intention to capture features for the word level semantic representation to counter the limitation of previous part of the architecture. For this purpose LSTM is used as well, because LSTM has performed very well (Bhasin, 2019; Tang, 2015) in various sentiment analysis and other text processing tasks.

Formally a Sentence S is made of sequence of words $[p_1, \dots, p_l]$ where l is word length of S. $Q \in \mathbb{R}^{d \times l}$ is the representation of sentence where d being the dimension of word embedding. Now p_t, word at time t is passed to memory cell of LSTM and the output follows similar of equation (2).

We have introduced two separate attention layers over the LSTM output of Character based side and Word based side respectively. The intention of applying the attention is to infer the dominating features from character representation as well as word representation respectively. We have used self attention (Vaswani, 2017) for our implementation[9]. Formally, the attention can be depicted as equation (3).

$$Attention(Q, K, V) = softmax(QK^T/d_k) \tag{3}$$

The Q, K & V is same and that is the output of the previous layer. The final output after attention of sub-word level representation through character embedding part and learnt features from the word embedding part are concatenated as late fusion to feature represent of the input sentence. The joint feature is passed through another attention layer. This layer is intend to figure out the dominating learnt feature among word and character based learnt features. Following this layer, we add two consecutive fully connected layers with ReLU non-linearity. The final output of the last dense layer is passed through a Softmax layer to predict the sentiment.

Formally late fusion of learnt character features f_c & word features f_w is $f_s = (f_c, f_w)$ to represent jointly learnt features of sentence S. Then $_s$ is input to dense layers with g as ReLU non-linearity. Output a_1 is passed through second dense layer to get output a_2.

$$\begin{aligned} a_1 &= g(W_1 \times f_s + b_1) \\ a_2 &= g(W_2 \times a_1 + b_2) \end{aligned} \tag{4}$$

Further, final layer is formalized as equation 5.

$$\sigma = e^{a_2} / \sum e^{a_{2_i}} \tag{5}$$

5 Experimental Setup

This research used Keras on python for all required implementations. The baseline dataset is divided into 3 splits i.e. training, validation and testing. Initially the dataset is randomly divided into 80-20 train-test split. Further train is randomly divided into 90-10 train-validation unlike the baseline implementation which splits 80-20 as train-validation. The results are reported over the test split here.

We have experimented with various possible values of hyper parameters and the best set of hyper parameters is shown in the Fig 2. As discussed earlier first part of the architecture is meant for character based input. Here a single

[9] https://pypi.org/project/keras-self-attention/

sentence is considered to be of sequence of 200 characters. Characters beyond 200 are ignored for sentence having more than 200 characters. A sentence with less than 200 characters is zero padded. Point need to mention is that we have considered space also as valid character input. For the second part of the network we have use word embedding of different dimensions for example 100, 200 and 300. However it achieved best accuracy with 300 dimensional word-embedding. While training the fastText Word-Embedding, 'minn' & 'maxn' parameters were set to 2 and 10 respectively. For word based input, a sentence of length 40 words is considered. A sentence with lesser than 40 words is zero embedding padded whereas words beyond 40 are ignored if sentence is having more than 40 words. Also we empirically found that having two stacked LSTM layers similar to Prabhu (2016) gave optimal performance.

We have used default Keras implementations of Categorical Cross Entropy for loss functions in different experiments. Available implementation of Focal Loss[10] is used during few experiments. The intention of apply focal loss (Lin, 2017) is to check the robustness of the proposed ACCMT architecture with respect to different loss function. Of late Focal Loss has migrated from Object Detection to various other tasks, for example speech emotion recognition Tripathi (2019) etc. We wanted to experiment and capture the impact of Focal Loss on Classification of Code-Mixed text. Default Keras implementation for adam optimizer is used for experiments. On the other hand learning rate of 0.0008 and a decay of 0.000012 is set for RMS Prop in various set of experiments. Dropout at Character-LSTM part is set to 0.2 and Word-LSTM is set to 0.4, where as the dropout of dense layers are set to 0.4. We have used the available implementation of attention layer in our code for model architecture.

The model is trained over 50 epochs and batch size of 64 with 10-fold cross-validation. During each fold, the best model is picked based on validation accuracy. The experiments are conducted in the Anaconda environment on a machine with Intel Core i5 processor and NVIDIA processor for GPU acceleration, 16 GB of RAM and a 1 TB of HDD with Windows 10 Operating System. The 50 epochs of training of ACCMT takes 25 minutes in average.

6 Results and Analysis

We have conducted all experiments in the computing environment mentioned in above section. In the same environment, the implementation of Prabhu (2016) attained maximum accuracy of 66.29% across 5 different executions. Whereas the best performance of ACCMT is 71.97% exceeds the baseline performance by 5.68% in the same computing environment. To understand the impact of attention on the classification of code-mixed text, we have also experimented without attention. We have removed three attention layers from the ACCMT and created a deep learning architecture which uses only fusion of character and word features. This architecture showed a maximum of 69.845% accuracy on the same dataset. This implies that attention has improved accuracy with 2.125%. We also compared against Yenigalla (2018) which gave an accuracy of 64.3%. Table 2 showed the accuracy and F1 score of all experiments.

Experiments	***Results***	
	Accuracy	***F1***
Yenigalla (2018)	64.3%	62.2
ACCMT (adamax + Focal Loss)	70.10%	68.1
ACCMT (RMS prop + categorical cross entropy)	69.75%	67.5
ACCMT (adamax + categorical cross entropy)	**71.97%**	**70.93**
ACCMT (RMS Prop + Focal Loss)	70.32%	68.71

Table 2: Results of ACCMT on Hi-En Code Mixed dataset with different loss-function and initializers.

7 Conclusion

This paper shows the architecture of attention based deep learning architecture (ACCMT) which does fusion of character and word feature to develop a robust classifier for code-mixed text. The proposed ACCMT architecture performs well on the Hi-En code-mixed dataset and outperforms the baseline accuracy. A major contribution of this paper is the technique of training word embedding for code-mixed text. This technique is used for

[10] https://github.com/mkocabas/focal-loss-keras

generating word embedding for Hindi-English code mixed corpus, which is required in this research work. This proposed technique is very easy to implement for other code-mixed languages as well and will be helpful for generating word embedding for low resource code-mixed languages majorly Indian languages e.g. Bengali, Tamil and Malayalam etc. This also opens up opportunities of research on other code-mixed languages. This work also shows the impact of attention for the classification of code-mixed text. Lal (2019) showed that introduction of feature network has improved the accuracy significantly. The integration of such feature network in ACCMT is considered for future course of improvement for the on-going research.

References

Patra, B.G., Das, D. and Das, A., 2018. *Sentiment Analysis of Code-Mixed Indian Languages: An Overview of SAIL_Code-Mixed Shared Task@ ICON-2017*. arXiv preprint arXiv:1803.06745.

Chanda, Arunavha, Dipankar Das, and Chandan Mazumdar. *Unraveling the English-Bengali code-mixing phenomenon.* In Proceedings of the Second Workshop on Computational Approaches to Code Switching, pp. 80-89. 2016.

Dey, A. and Fung, P., 2014, May. *A Hindi-English Code-Switching Corpus*. In LREC (pp. 2410-2413)

Barman, U., Das, A., Wagner, J. and Foster, J., 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 (pp. 13-23).

Prabhu, A., Joshi, A., Shrivastava, M. and Varma, V., 2016. *Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text*. arXiv preprint arXiv:1611.00472.

Yenigalla, P., Kar, S., Singh, C., Nagar, A., & Mathur, G. (2018, June). *Addressing unseen word problem in text classification*. In International Conference on Applications of Natural Language to Information Systems (pp. 339-351). Springer, Cham.

Lal, Y.K., Kumar, V., Dhar, M., Shrivastava, M. and Koehn, P., 2019, July. *De-Mixing Sentiment from Code-Mixed Text*. In Proceedings of the 57th Conference of the Association for Computational Linguistics: Student Research Workshop (pp. 371-377).

Bhasin, A., Natarajan, B., Mathur, G., Jeon, J.H. and Kim, J.S., 2019, June. *Unified Parallel Intent and Slot Prediction with Cross Fusion and Slot Masking*. In International Conference on Applications of Natural Language to Information Systems (pp. 277-285). Springer, Cham.

Sharma, S., Srinivas, P. Y. K. L., & Balabantaray, R. C. (2015, August). *Text normalization of code mix and sentiment analysis.* In 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 1468-1473). IEEE.

Pravalika, A., Oza, V., Meghana, N.P. and Kamath, S.S., 2017, July. *Domain-specific sentiment analysis approaches for code-mixed social network data*. In 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). *Enriching word vectors with subword information*. Transactions of the Association for Computational Linguistics, 5, 135-146.

Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016, March). *Character-aware neural language models* In Thirtieth AAAI Conference on Artificial Intelligence.

Tang, D., Qin, B., & Liu, T. (2015, September). *Document modeling with gated recurrent neural network for sentiment classification*. In Proceedings of the 2015 conference on empirical methods in natural language processing (pp. 1422-1432).

Smith, S.L., Turban, D.H., Hamblin, S. and Hammerla, N.Y., 2017. *Offline bilingual word vectors, orthogonal transformations and the inverted softmax*. arXiv preprint arXiv:1702.03859.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C. and Joulin, A., 2018, May. *Advances in Pre-Training Distributed Word Representations*. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. *Attention is all you need*. In Advances in neural information processing systems (pp. 5998-6008).

Lin, T.Y., Goyal, P., Girshick, R., He, K. and Dollár, P., 2017. Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).

Tripathi, S., Kumar, A., Ramesh, A., Singh, C. and Yenigalla, P., 2019. *Focal Loss based Residual Convolutional Neural Network for Speech Emotion Recognition*. arXiv preprint arXiv:1906.05682.

Dataset for Aspect Detection on mobile reviews in Hindi

Ayush Joshi Pruthwik Mishra Dipti Misra Sharma
Language Technologies Research Center
Kohli Center On Intelligent Systems, IIIT Hyderabad
{ayush.joshi, pruthwik.mishra}@research.iiit.ac.in,,dipti@iiit.ac.in

Abstract

In recent years Opinion Mining has become one of the very interesting fields of Language Processing. To extract the gist of a sentence in a shorter and efficient manner is what opinion mining provides. In this paper we focus on detecting aspects for a particular domain. While relevant research work has been done in aspect detection in resource rich languages like English, we are trying to do the same in a relatively resource poor Hindi language. Here we present a corpus of mobile reviews which are labelled with carefully curated aspects. The motivation behind Aspect detection is to get information on a finer level about the data. In this paper we identify all aspects related to the gadget which are present on the reviews given online on various websites. We also propose baseline models to detect aspects in Hindi text after conducting various experiments.

1 Introduction

Over the last decade people tend to search for products online rather than physically on stores. This has resulted in a surge of online forums where reviews are available on various products, electronic gadgets being one of the more popular ones. But reading so many long reviews is very time consuming and there is no uniformity on the parameters of reviews. To solve this, research work has been done in this area in the form of Aspect Detection which helps to point out the key specifications of the product in a structured format. But the work is limited to only worldwide languages as English and French. For a multi-lingual country like India, we are still far away in getting these information in the native language. We aimed at creating a dataset in Hindi which has the highest number of native speakers. The dataset is annotated with aspects for mobile reviews.

An aspect is a word in a sentence which has some polarity associated with it. The aspect should hold major meaning of the sentence. Following examples will state what aspect is:

S1 : शाओमी रेडमी 4ए को पहली बार हाथ में लेने पर यह आपको मेटल बॉडी का बना लगेगा ।
S1 : Xiaomi redmi 4A ko pehli baar hath m lene par yeh aapko metal body ka bana lagega.

Aspect1 : "मेटल बॉडी" (metal body) which falls under the "डिज़ाइन" (design) category. The aspect shows importance by indicating how the mobile is built.

S2: शाओमी रेडमी नोट में 2 गीगाहर्ट्ज़ ऑक्टा – कोर क्वालकॉम स्नैपड्रैगन 625 प्रोसेसर का इस्तेमाल हुआ है।
S2: Xiaomi Redmi note m 2 gigahertz octa-core qualcomm snapdragon 625 processor ka istemal hua hai. Aspect:"ऑक्टा – कोर क्वालकॉम स्नैपड्रैगन 625" (Octa core qualcomm snapdragon) which falls under the "स्पेसिफिकेशन" (specification) category . The aspect tells specifically tells the details of product.

S3: अफसोस यह कि आप उन्हें हटा नहीं सकते ।
S3: Afsos yeh ki aap unhe hata nahi sakte.
Aspect : "NULL" as there is no word which tells about any detail of the product. Hence, it is classified under no aspect category.

2 Related Work

Major work has been done in Aspect Detection when it comes to resource rich languages like English. The work of Aspect Detection has also been followed by Sentiments analysis which plays a major part in Opinion mining. In 2014 SemEval-Task 4, Maria Pontiki (2014) provided the first dataset which con-

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 130–134
Hyderabad, India, December 2019.

Aspect Class In Hindi	Aspect Class In Roman	Count
सॉफ्टवेयर	software	52
स्पेसिफिकेशन और फ़ीचर	Specification aur feature	360
हमारा फ़ैसला	hamara faisla	9
कैमरा और बैटरी लाइफ	camera aur battery life	5
स्पेसिफिकेशन और सॉफ्टवेयर	specification aur software	137
कैमरा	camera	76
परफॉर्मेंस	performance	826
लुक व बनावट	look vah banawat	26
बैटरी लाइफ	battery life	1
हमारा फैसला	hamara faisla	300
कैमरा परफॉर्मेंस	camera performance	16
डिज़ाइन	design	138
डिज़ाइन और लुक	design aur look	168
NULL	NULL	352
स्पेसिफिकेशन	specification	139
डिज़ाइन और बिल्ड	design aur build	390
डिज़ाइन और डिस्प्ले	design aur display	49
स्पेसिफिकेशन , सॉफ्टवेयर और परफॉर्मेंस	specification, software aur performance	40

Table 1: Class Set

sisted of English reviews annotated at sentence level with their aspects followed by their polarity. Some of the systems that emerged who targeted this task were Zhiqiang Toh (2014), Chernyshevich (2014); Joachim Wagner and Tounsi (2014); Giuseppe Castellucci (2014), Shweta Yadav (2015). However, almost all these systems are related to some specific languages, especially English. In 2016, SemEval released new datasets of similar domains(mobile, laptop, restaurant) [1] but in multiple languages. In 2016, the datasets were released in English, Arabic, Chinese, Dutch, French, Russian, Spanish and Turkish.

But this area of field is largely unexplored in Indian languages due to the unavailability of high quality datasets and other tools and resources required. The datasets which were created by research groups mainly by Aditya Joshi (2010); Balamurali A R (2011, 2012) were very less in size and low in quality. Also Google transolator was used to create data in Indian languages (Akshat Bakliwal, 2012) but dataset created was not rich enough to perform aspect detection with high efficiency. Moreover the datasets available in Hindi were not domain specific which also added to poor results in past.

3 Data Creation

As mentioned, earlier our work is on a specific domain. To build our corpus we scrapped data from various online forums with reviews on mobile phones. We extracted the text from the HTML data with the help of BeautifulSoup library [2] in python. As our language was Hindi, online reviews were very less for which we tried both dynamic and manual crawling of data.

After crawling over 8 websites, we were able to get over 381 reviews. We retrieved 294 mobile reviews(37410 sentences) in a HTML format after extensive removal of noisy reviews. We had 294 HTML files which had raw data between different HTML tags. There was no uniformity in the reviews, even after extraction and tokenization of these reviews,

[1] http://alt.qcri.org/semeval2016/task5/index.php?id=data-and-tools

[2] https://pypi.org/project/beautifulsoup4/

Unclean reviews	381
Unclean sentences	37410
Clean reviews	294
Clean sentences	2000
Total tokens	34359

Table 2: Corpus Details

Many reviews had proper headings like specifications, performance, price, design under which two-three paragraphs of text was present. But there were many reviews without any headings. To make it uniform and bring it to sentence level rather than paragraph level, we assigned the heading as labels to every sentence appearing under that heading in the review. This was our first annotation strategy. While assigning heading as aspects, there were certain sentences which had no heading above them. Such sentences were labelled as NULL. After this initial annotation, we had 18 classes of aspects in total. After doing analysis on our 18 classes, we observed a lot of overlapping between different classes. Some classes had the same name, but due to spelling variations they were assigned different labels. Table 3 gives a clear picture about the overlapping between different classes. We show the counts of highly frequent overlapping class pairs.

Class1 and Class2	**Overlap Count**
स्पेसिफिकेशन और फ़ीचर, स्पेसिफिकेशन और सॉफ्टवेयर *(specification aur feature), (specification aur software)*	441
डिज़ाइन और लुक, डिज़ाइन और बिल्ड *(design aur look), (design aur build)*	418
स्पेसिफिकेशन और फ़ीचर, स्पेसिफिकेशन *(specification aur feature, specification)*	410
डिज़ाइन, डिज़ाइन और बिल्ड *(design), (design aur build)*	387
स्पेसिफिकेशन स्पेसिफिकेशन और सॉफ्टवेयर *(specification), (specification aur software)*	336

Table 3: Overlapping Between Initial Classes

The following decisions to club different classes and provide them a single label were taken based on the percentage of overlapping.

- सॉफ्टवेयर*(software)*, स्पेसिफिकेशन और फ़ीचर *(specification aur feature)*, स्पेसिफिकेशन और सॉफ्टवेयर *(specification aur software)*, स्पेसिफिकेशन, स्पेसिफिकेशन , सॉफ्टवेयर और परफॉर्मेंस *(specification, specification, software aur perfomance)* clubbed under one single class called स्पेसिफिकेशन *(specification)*.
- कैमरा और बैटरी लाइफ *(camera aur battery life)*, कैमरा *(camera)*, , कैमरा परफॉर्मेंस *(camera performance)* were clubbed under a class कैमरा *(camera)*.
- लुक व बनावट *(look wh banawat)*, डिज़ाइन *(design)*, डिज़ाइन और लुक *(design aur look)*, डिज़ाइन और बिल्ड *(design aur build)*, डिज़ाइन और डिस्प्ले *(design aur display)* categorized under one class डिज़ाइन*(design)*.
- कैमरा *(camera)*,कैमरा परफॉर्मेंस *(camera performance)*, कैमरा और बैटरी लाइफ*(camera aur battery life)* were categorized under one class कैमरा *(camera)*.
- NULL and हमारा फैसला*(hamara faisla)* were merged as into a single class NULL.

After eliminating all these redundancies, we finally had 5 classes or aspects for our mobile reviews.

Aspect Class	**Count**
डिज़ाइन *(design)*	298
स्पेसिफिकेशन*(specification)*	585
NULL	489
परफॉर्मेंस*(performance)*	459
कैमरा*(camera)*	169

Table 4: Classwise Distribution

Two annotators were involved in this task. We obtained a Fleiss' [3] score of 0.87 for inter annotator agreement.

4 Experimental Setup

The main task was to predict aspects in every sentence in a review. We used different

[3] https://en.wikipedia.org/wiki/Fleiss'_kappa

Classifier	Feature	P	R	F1-Score
MNB	word uni	0.65	0.60	0.62
MNB	word uni+bi	0.62	0.63	0.63
MNB	char 2gram	0.72	0.65	0.67
MNB	char 2-3gram	0.75	0.73	0.74
MNB	char 2-4gram	0.74	0.74	0.74
MNB	char 2-5gram	0.73	0.75	0.74
MNB	word uni+char2-5gram	0.74	0.74	0.74
MNB	word uni+bi+char2-5gram	0.73	0.75	0.74
SVM	word uni	0.65	0.64	0.65
SVM	word uni+bi	0.70	0.66	0.67
SVM	char2gram	0.73	0.71	0.72
SVM	char2-3gram	0.75	0.73	0.74
SVM	char2-4gram	0.77	0.75	0.75
SVM	char2-5gram	0.77	0.75	0.76
SVM	word uni+char2-5gram	0.74	0.73	0.73
SVM	word uni+bi+char2-5gram	0.75	0.73	0.74

Table 5: Results Of Models After 5-fold Cross Validation

classifiers for the prediction task. We mostly experimented with machine learning models with 5-fold cross-validation as we had limited amount of data at our disposal.

4.1 Feature Engineering

Feature engineering is critical in designing accurate models. The features used in designing our supervised learning models are detailed here.

TF-IDF Vectors

- Word n-grams - This feature deals with the presence or absence of certain sequence of words. The value of n used varied from 1 to 2.
- Character n-grams - This is similar to word n-grams where a sequence of characters is extracted from the text. The value of n used varied from 2 to 5.

4.2 Machine Learning Approach

We created baseline with two classifiers

- Support Vector Machines (SVM)
- Multinomial Naive Bayes (MNB)

These two classifiers were implemented using the sklearn (Pedregosa et al., 2011) library. We used different feature set in both the classifiers.

5 Results

The results are shown in table 5. Classifiers and their corresponding features are detailed in this table. We used precision, recall and macro F1-score as the evaluation metric for checking the performance of our models. The words 'uni', 'bi' refer to the word unigrams and bigrams respectively. char 'a-b' gram denotes the combination of character n-grams where n lies in $\{a, a+1, a+2, .., b\}$

6 Observation

From table 5, we observed that both the classifiers equally perform well on the data. We also observed that character n-grams models are superior than word n-gram models. Combination of word and char n-gram TF-IDF vectors do not significantly improve the performance.

From the values of confusion matrix, we observed that class स्पेसिफिकेशन*(specification)* has overshadowed classes NULL and कैमरा*(camera)*. It shows that our model is not able to predict between the umbrella class and the child class accurately.

7 Conclusion and Future Work

We annotated aspects for mobile reviews written in Hindi as a part of this work. We also presented baseline models for automatic aspect identification in mobile reviews. The

baseline models will help us to annotate more reviews semi-automatically and can then be integrated to improve our systems. We will explore more into neural network architecture and word embeddings. The next task in this area would be to annotate polarity of the aspects. We can also explore identifying the most informative reviews (Mishra et al., 2017).

References

Pushpak Bhattacharyya Aditya Joshi, Balamurali A R. 2010. A fall-back strategy for sentiment analysis in hindi: a case study. In *Proceedings of ICON 2010: 8th International Conference on Natural Language Processing, Macmillan Publishers, India.*

Vasudeva Varma Akshat Bakliwal, Piyush Arora. 2012. Hindi subjective lexicon : A lexical resource for hindi polarity classification. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC).*

Aditya Joshi Pushpak Bhattacharyya Balamurali A R. 2011. Harnessing wordnet senses for supervised sentiment classification. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, Edinburgh, Scotland, UK, July 27–31.*, pages 1081–1091.

Aditya Joshi Pushpak Bhattacharyya Balamurali A R. 2012. Cross-lingual sentiment analysis for indian languages using linked wordnets. In *Proceedings of COLING 2012: Posters, COLING 2012, Mumbai, December 2012.*, pages 73–82.

Maryna Chernyshevich. 2014. Ihs rd belarus: Cross-domain extraction of product features using conditional random fields. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014) .*, pages 309–313.

Danilo Croce Roberto Basili Giuseppe Castellucci, Simone Filice. 2014. Unitor: Aspect based sentiment analysis with structured learning. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), Dublin, Ireland .*, pages 761–767.

Santiago Cortes Utsab Barman Dasha Bogdanova Jennifer Foster Joachim Wagner, Piyush Arora and Lamia Tounsi. 2014. Dcu: Aspect-based polarity classification for semeval task 4. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), Dublin, Ireland .*, pages 223–229.

John Pavlopoulos Haris Papageorgiou Ion Androutsopoulos Suresh Manandhar Maria Pontiki, Dimitrios Galanis. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014).*, pages 27–35.

Pruthwik Mishra, Prathyusha Danda, Silpa Kanneganti, and Soujanya Lanka. 2017. Iiit-h at ijcnlp-2017 task 3: A bidirectional-lstm approach for review opinion diversification. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 53–58.

Fabian Pedregosa, Gäel Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Sriparna Saha Shweta Yadav, Asif Ekbal. 2015. Feature selection for entity extraction from multiple biomedical corpora: A pso-based approach. In *In Natural Language Processing and Information Systems, Springer.*, pages 220–233.

Wenting Wang Zhiqiang Toh. 2014. Dlirec: Aspect term extraction and term polarity classification system. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), Dublin, Ireland, August 23-24*, pages 235–240.

Multi-linguality Helps: Event-Argument Extraction for Disaster Domain in Cross-lingual and Multi-lingual Setting

Zishan Ahmad, Deeksha Varshney, Asif Ekbal, Pushpak Bhattacharyya
Department of Computer Science and Engineering
Indian Institute of Technology Patna
{1821cs18, 1821cs13, asif, pb}@iitp.ac.in

Abstract

Automatic extraction of disaster-related events and their arguments from natural language text is vital for building a decision support system for crisis management. Event extraction from various news sources is a well-explored area for this objective. However, extracting events alone, without any context provides only partial help for this purpose. Extracting related arguments like *Time*, *Place*, *Casualties*, etc., provides a complete picture of the disaster event. In this paper, we create a disaster domain dataset in *Hindi* by annotating disaster-related event and arguments. We also obtain equivalent datasets for *Bengali* and *English* from a collaboration. We build a multi-lingual deep learning model for argument extraction in all the three languages. We also compare our multi-lingual system with a similar baseline monolingual system trained for each language separately. It is observed that a single multi-lingual system is able to compensate for lack of training data, by using joint training of dataset from different languages in shared space, thus giving a better overall result.

1 Introduction

The ability to extract real time news of disaster events automatically, can potentially help in better decision-making for planning and coordination of disaster relief efforts. Event extraction from text entails the extraction of particular types of events along with their arguments. Information obtained from extracted event mentions provides a more structured and clear picture when augmented with related arguments like *Time*, *Place*, *Participant*, *Casualty* etc. In a language rich world where each event is documented in multiple languages, argument extraction in multi-lingual setting stands as a crucial task.

Extraction of events from news is a well explored area in Natural Language Processing. Competitions such as ACE2005 (Doddington et al., 2004) and TAC-KBP2015 (Mitamura et al., 2015) have investigated the area and provided a large body of literature on event extraction from news articles. Event extraction was done on ACE2005 dataset by Ji and Grishman (2008) by combining global evidence from related documents with local decisions. Hou et al. (2012) introduced a method of event argument extraction based on CRFs model for ACE 2005 Chinese event corpus. Event and its arguments were extracted by Petroni et al. (2018), for the purpose of extracting breaking news. Although extraction of events is quite well examined, there is a scarcity of work in extraction of detailed arguments for disaster domain like *casualties*, *reason*, *after-effects* etc.

In this paper we create and publish a dataset annotated for events in disaster domain, for three different languages, i). *Hindi*, ii). *Bengali* and iii). *English*. This dataset is annotated for the task of argument extraction by expert annotators. We build a 'mono-lingual' deep learning system, based on CNN (Convolutional Neural Network) and Bi-LSTM (Bi-Directional Long Short Term Memory) for the task of argument extraction. In order to leverage the information from all the languages while training, and improve the performance of the system, we build a 'multi-lingual' argument extraction system. This is done by adding separate language layers for each language to our 'mono-lingual' system. To bring the datasets of all the languages to the same vector space, we make use of 'multi-lingual' word embeddings. We show that by training our model in this way we are able to utilize the dataset of all the three languages and improve the performance of our system for most arguments in the three languages. We also investigate how the syntactic difference of the languages is handled by our system. Through analysis, we show that 'multi-lingual' training is espe-

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 135–142
Hyderabad, India, December 2019.

cially helpful in improving the performance when some argument is under-represented in the 'monolingual' training data.

1.1 Problem Definition

Argument extraction entails classifying each word in the sentence into some argument or not argument. Therefore, it has been formulated as a sequence labelling task. Given a sentence of form $w_1, w_2, ..., w_n$, the task is to predict the sequence of event-arguments, of the form $l_1, l_2, ..., l_n$. Six different types of arguments were annotated in the dataset: i). *Place*, ii). *Time*, iii). *Reason*, iv). *Casualties*, v). *Participant* and vi). *After-effects*. To label multi-word event-arguments, IOB-style encoding is used where B, I and O denote the beginning, intermediate and outside token of an event.

- **Input Hindi Sentence:** गृह मंत्रालय **मुंबई** के बम विफोटों के मद्देनजर इस बात की विशेष तौर पर जांच कर रहा है कि **अक्षरधाम मंदिर** और **१९९३** के **मुंबई** बम विस्फोटों के फसलों की प्रतिक्रिया के रूप में तो यह हमले नहीं हुए
- **Translation**: In view of the **Mumbai** bomb blasts, the Home Ministry is specially investigating the fact that these attacks did not take place as response to the **Akshardham Temple** and the **1993 Bombay** bomb blasts
- **Output**: O O I_Place O O O O O O O O O O O O O O I_Place I_Place O I_Time O I_Place O O O O O O O O O O O O O O

2 Related Works

A major task in information extraction is detection of event triggers, event classification and event argument extraction. Recent works on event trigger detection and classification discuss efficient feature representation techniques which can help in event extraction. Nguyen and Grishman (2015) proposed a convolutional neural network for event extraction which automatically learns features from text. Chen et al. (2015) introduced dynamic convolutional neural network (DMCNN), which adopt a dynamic multi-pooling layer in accordance with the event triggers and its arguments. In 2016, Nguyen and Grishman (2016) improved their CNN model by introducing the non-consecutive convolution by skipping irrelevant words in a sequence. Feng et al. (2018) designed a combined model of LSTM's and CNN's which helped in capturing both sequence level and chunk level information from specific contexts. Nguyen and Grishman (2018) explored graph convolutional network over dependency trees and entity mention-guided pooling. For low resource languages, Liu et al. (2018) came up with Gated Multi-Lingual Attention (GMLATT) and Lin et al. (2018) developed a multi-lingual multi-task architecture alleviating data sparsity problem in related tasks and languages.

Previously, in event argument extraction researchers have experimented with pattern based methods (Patwardhan and Riloff, 2007; Chambers and Jurafsky, 2011) and machine learning based methods (Patwardhan and Riloff, 2009; Lu and Roth, 2012) most of which utilise the various kinds of features obtained from the context of a sentence. Higher level representations such as cross-sentence or cross-event information were also explored by Hong et al. (2011) and Huang and Riloff (2011). Maximum Entropy based classifiers were applied for event and argument labeling by Ahn (2006); Chen and Ji (2009); Zhao et al. (2008). The disadvantage with ME classifier is that it gets stuck in local optima and fails to fully capture the context features. To overcome this Hou et al. (2012) proposes a event argument extraction system based on Conditional Random Fields (CRF) model that can select any features and normalizing these features in overall situation helps in obtaining optimal results. While, these models can get affected by the error propagated from upstream tasks, a joint model can help us utilise the close interaction between one or more similar tasks. Li et al. (2013) presented a joint model for Chinese Corpus which identifies arguments and determines their roles for event extraction using various kinds of discourse-level information. On ACE2005 dataset Sha et al. (2018) proposed a dependency bridge recurrent neural network (dbRNN) built upon LSTM units for event extraction. They use dependency bridges over Bi-LSTM to join syntactically similar words. A tensor layer is applied to get the various argument-argument interactions. Event triggers and arguments are then jointly extracted utilising a max-margin criterion. Nguyen et al. (2016) presented a GRU model to jointly predict events and its arguments.

We introduce two systems for the task of event argument extraction. First is our monolingual system built using CNN (Convolutional Neural Network) and Bi-LSTM (Bi-Directional Long Short

Term Memory). To exploit the information from related languages, we develop a second system that can use information from all the languages for training. This multi-lingual system is built by using shared vector space of embeddings while training, and by using separate language layers for each language to accommodate for diversity in syntax of the languages.

3 Methodology

In this paper, we propose that joint training of IE system on different language datasets, using 'multi-lingual' word embeddings and language layers helps in better extraction of arguments. This is particularly true when the dataset is limited in size. To corroborate our claim, we device two different systems, i). monolingual baseline system, and ii). multi-lingual system. The 'monolingual baseline' system only takes input data (sentence wise) from one language and extracts the arguments. For word representation, it uses monolingual word embeddings. The 'multi-lingual' argument extraction system uses separate language layers and multi-lingual word embeddings for joint training on all the three languages.

3.0.1 Monolingual Word Embedding

The monolingual word-embeddings that are used in our experiments are also known as fastText[1]. It was proposed by Bojanowski et al. (2017), and is based on the skipgram model. However instead of using one-hot vector encoding for each word while training, a vector representation of a word that considers character n-grams occurring in the word is formed. To get this representation, the n-grams from all the words for 'n' greater than 2 and smaller than 7 are extracted. After this, a dictionary of all the extracted n-grams is created. A given word w, can now be denoted by $\Gamma_w \subset \{1, ..., G\}$ i.e the set of n-grams appearing in the word; where G is the size of the n-gram dictionary. With each n-gram in G, a vector representation z_g is associated. A word represention is obtained by summing up all the n-grams, as described in Equation 1:

$$V_w = \sum_{g \in G_w} z_g \tag{1}$$

The continuous skip-gram model used these word vectors V_w, to obtain word-embedding representations of words. The main advantage of this technique is that, even in the absence of some word in the training corpus, some representations of the word is still obtained as the n-gram representation of words is considered. This skip-gram model is trained using *Wikipedia* data dump of each language. The dimension of the word vector to is set to 300.

3.0.2 Multi-lingual Word Embedding

Multi-lingual embeddings are obtained by learning a mapping matrix W, between source embeddings $X = \{x_1, x_2, x_3..., x_n\}$ and target embeddings $Y = \{y_1, y_2, y_3, ..., y_n\}$ without cross-lingual supervision.Adversarial training was used in this method proposed by Conneau et al. (2017). A discriminator is trained to discriminate between a randomly sampled element from $WX = \{Wx_1, ..., Wx_n\}$ and Y. At the same time W is trained to prevent the discriminator from making correct prediction. Thus making it a two-player game, where the discriminator tries to maximize its capability of identifying the origins of an embedding, and W aims to prevent the discriminator from doing so by making WX and Y as indistinguishable as possible. The W matrix is trained with near orthogonality constraint, to ensure that while transforming the source vector to the target vector space, the angles and distances between words in the embeddings are not distorted during transformation. To achieve this near orthogonality constraint, weight updation for W is done using Equation 2.

$$W \leftarrow (1 + \beta)W - \beta(WW^T)W \tag{2}$$

Here, β was set to 0.01 for the transformation. For our experiments we trained mapping matrices W_{hindi} and $W_{bengali}$ that map the *Hindi* and *Bengali* word embeddings to the vector space of *English* embeddings.

3.1 Monolingual Baseline Model

The 'monolingual baseline' model (c.f Figure 1) is based on Bi-Directional Long Short Term Memory (Bi-LSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) and Convolutional Neural Networks (CNN) (Kim, 2014). The input to the model is a sentence, represented by a sequence of monolingual word embeddings. Since Bi-LSTM and CNN take sequences of equal lengths, the shorter sequences are padded by zero

[1] https://github.com/facebookresearch/fastText

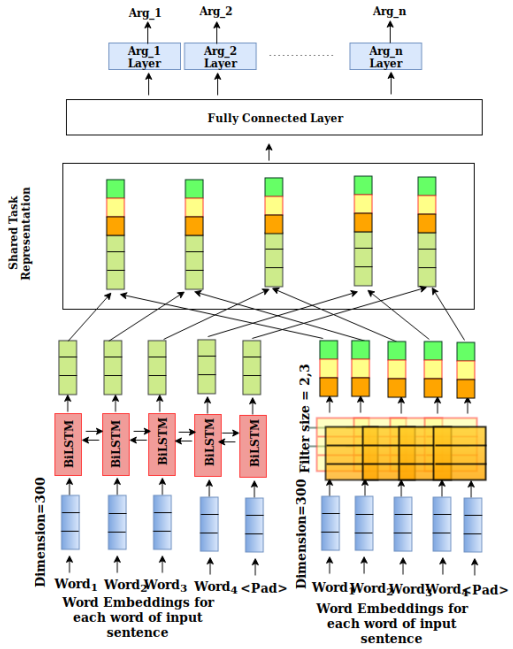


Figure 1: monolingual baseline model for argument extraction

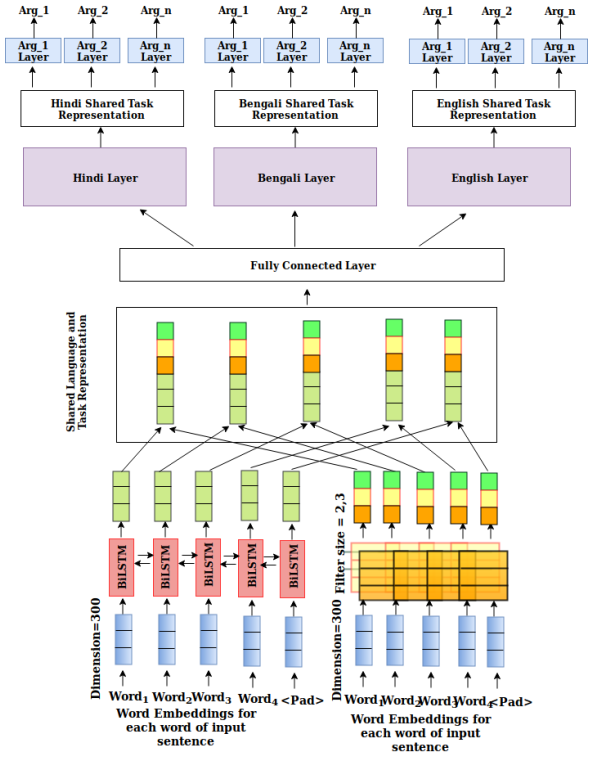


Figure 2: Multi-lingual baseline model for argument extraction

vectors. This sequence is passed through Bi-LSTM and CNN having filter size 2 and 3. The Bi-LSTM gives contextual representation of each word, while the CNN extracts the 'bi-gram' and 'tri-gram' features for the sequence. These features are concatenated and passed through a fully connected layer. This layer gives shared representation for the task of argument extraction. Since the arguments in the dataset are not mutually exclusive (E.g: *Place* or *Participant* argument can also be a part of *Reason* or *After-effect* argument), we have different layers to predict different arguments independently. We have 6 different fully-connected layers in parallel, each of them specialized for detection of one of the 6 arguments. 'Softmax' is used after each of the final layers to classify the representation into I, O or B of an argument.

3.2 Multi-lingual Model

For multi-lingual system, we build a model based on the baseline model, by adding separate language layers (L_1, L_2 and L_3) for each language (c.f Figure 2). A layer L_i and its subsequent layers are only trained when input data is also of language L_i. We represent the input sentence as a sequence of multi-lingual word embeddings, and padding with zero vectors is used to make the sequence equal in length. Similar to the 'monolingual baseline' model, Bi-LSTM, CNN and a fully connected layer is used. This fully connected layer produces shared language and task representation as output. Three separate language layers for the languages *Hindi*, *Bengali* and *English* are used in parallel. These language layers decode the language specific representation from shared representation. After each language layer we have 6 fully connected layers for each of the 6 arguments. 'Softmax' classifier is used to classify the representation into I, O or B of an argument.

4 Dataset and Experiments

In this section, we describe the dataset used and the experiments conducted.

4.1 Dataset

To create the dataset, we crawled news articles in disaster domain from popular news websites in *Hindi*. These news articles were annotated by three annotators, with good language abilities and having satisfactory knowledge in the relevant area. The guidelines for annotation used were similar to the guidelines given by TAC KBP 2017 Event Sequence Annotation Guidelines[2]. We recorded that the annotators had Kappa agreement score of 0.85

[2] https://cairo.lti.cs.cmu.edu/kbp/2017/event/TAC_KBP_2017_Event_Coreference_and_Sequence_Annotation_Guidelines_v1.1.pdf

Argument	Hindi	Bengali	English
Time	3,953	11,042	822
Place	12,410	10,576	3,018
Reason	1,573	1,744	544
Casualties	12,171	15,870	4,823
Participant	2,264	4,311	639
After-effects	13,355	9,731	274

Table 1: Distribution of number of arguments in *Hindi*, *Bengali* and *English* datasets

on average. We also obtained equivalent dataset in *Bengali* and *English* language from a collaboration. The total dataset is comprised of 2,191 documents (*Hindi*: 922, *Bengali*: 999 and *English*: 270). It contains 44,615 sentences (*Hindi*: 17,116, *Bengali*: 25,717 and *English*: 1,782). The six arguments in the dataset and their distribution in the three languages are detailed in the Table 1.

4.2 Experiments

We conduct two separate experiments to show that dataset from different languages (L_1 and L_2) can be leveraged to improve the performance of argument extraction system of a different language (L_3). First we conduct experiment to obtain baseline results on 'mono-lingual' setup. Next, we perform experiment using the combined dataset of all the three languages using 'multi-lingual' argument extraction model.

4.2.1 Monolingual Experiment

This experiment is conducted separately on each dataset using the 'monolingual baseline model' (c.f. Figure 1) and monolingual *fastText* embeddings. The results of this experiment is used as a baseline, against which the results of the other experiment is compared. The following set-up is used for the experiment: i). learning rate: 1×10^{-2}, ii). batch size: 32, iii). optimizer: Adam (Kingma and Ba, 2014), iv). loss function: Binary cross-entropy. The best model based on validation-set accuracy was saved after 100 epochs.

4.2.2 Multi-lingual Experiment

This experiment is conducted on the combined dataset of three languages, using the 'multi-lingual model' (c.f Figure 2). Multi-lingual word embeddings (described in Section 3.2) were used for word representation in all the three languages, in this experiment. The same experimental set-up used for the 'monolingual baseline' experiment, is also used for this experiment. The training of multilingual system was done batch wise, i.e. each language branch was trained for one batch alternatively. The number of steps per epochs was decided by the number of batches needed to complete one epoch of the largest training set, among the different language datasets.

5 Results and Analysis

In this section, we discuss the results obtained for the two experiments described in Section 4.2. We also provide analysis of the results. F1-Score is used as an evaluation metric, and all the results reported are 5-Fold cross-validated. The results for both, 'monolingual' and 'cross-lingual' experiments are reported in Table 2. From the results, it can be observed that F1-score for *Hindi* and *English* datasets improve for most arguments (5 out of 6 arguments), while the results for *Bengali* dataset improves for three out of the six arguments.

We also test the statistical significance of each increment in F1-Score for argument extraction. The 'p-values' obtained after 't-test' are shown in Table 3. It can be seen that most improvements in F1-score are statistically significant.

It is observed that multi-word *Time* arguments are better captured by 'multi-lingual' model than by the 'monolingual baseline' model. An example of this can be seen in the following sentence:

- **Hindi Text:** एसएसपी संतोष कुमार सिंह ने बताया कि रविवार रात को जलालपुर पर तैनात पुलिसकर्मियों ने बाइक पर सवार दो युवकों को रोकने की कोशिश की
- **Transliteration:** esesapee santosh kumaar sinh ne bataaya ki **ravivaar raat** ko jalaalapur par tainaat pulisakarmiyon ne baik par savaar do yuvakon ko rokane kee koshish kee
- **Translation:** SSP Santosh Kumar Singh said that on **Sunday night**, policemen stationed at Jalalpur tried to stop two youths riding on bikes.

In the aforementioned sentence the actual phrase denoting time is 'रविवार रात' (Sunday night). However the 'monolingual' model only detects 'रविवार' (Sunday) as the *Time* argument. However, after multi-lingual training the entire time phrase is correctly detected. This is because the lack of training data for multi-word time arguments in *Hindi*,

	Mono-lingual			Multi-lingual		
Argument	**Hindi**	**Bengali**	**English**	**Hindi**	**Bengali**	**English**
Time	0.60	**0.86**	0.56	**0.61**	0.85	**0.58**
Place	**0.58**	**0.61**	**0.57**	0.56	0.59	0.55
Reason	0.01	0.19	0.14	**0.16**	**0.22**	**0.20**
Casualties	0.58	**0.73**	0.62	**0.59**	0.71	**0.63**
Participant	0.35	0.50	0.30	**0.41**	**0.53**	**0.32**
After-effects	0.25	0.28	0	**0.30**	**0.35**	**0.13**

Table 2: Results (F1-Scores) for 'mono-lingual' and 'multi-lingual' experiments on *Hindi*, *Bengali* and *English* datasets: 5-Fold cross-validated

Argument	**Hindi**	**Bengali**	**English**
Time	0.46	n/a	0.03
Place	n/a	n/a	n/a
Reason	0.03	0.18	0.04
Casualties	0.39	n/a	0.10
Participant	0.01	0.11	0.54
After-effects	0.04	0.09	0.01

Table 3: The 'p-values' obtained for each improvement in results from the baseline 'mono-lingual' to 'multi-lingual' experiment (n/a is used for instances where no improvement was observed)

is supplemented by training data from *Bengali* and *English*.

Another interesting observation is that, for *Casualty* argument of *English* dataset, the 'monolingual' system often confuses people as casualties, even when they are not. An example of such observation is as follows:

- **Actual:** Over 200000 people in 36 villages located 6 miles (10 km) from the volcano were advised to evacuate immediately.
- **Monolingual Prediction:** Over **200000 people** in 36 villages located 6 miles (10 km) from the volcano were advised to evacuate immediately.
- **Multi-lingual Prediction:** Over 200000 people in 36 villages located 6 miles (10 km) from the volcano were advised to evacuate immediately.

In the above example the phrase '200000 people' does not denote casualty, however the 'monolingual' model confuses it as casualty. This is due to the lack of training data in *English* to learn the difference between some count of people and actual casualty. However, after 'multi-lingual' training the model is able to make this distinction correctly.

The F1-score for *Place* arguments for all the datasets, is better for the 'monolingual baseline' model. This is because *Place* argument is present in good numbers for all the datasets, therefore there are enough instances for proper training of deep learning model, even in monolingual setting. Using 'multi-lingual model' for such cases is of little help. Furthermore, the syntactic difference between languages confuses the system, thus degrading the performance of the 'multi-lingual' system. A good example of this phenomenon is show below:

- **Actual:** Three youths lost their lives when the car they were travelling in collided with a truck near **Gaddoli village of Naraingarh in Ambala**.
- **Monolingual Prediction:** Three youths lost their lives when the car they were travelling in collided with a truck near **Gaddoli village of Naraingarh in Ambala**.
- **Multi-lingual Prediction:** Three youths lost their lives when the car they were travelling in collided with a truck near **Gaddoli village of Naraingarh** in **Ambala**.

It can be observed that the 'monolingual baseline' model predicts the entire phrase describing the *Place* argument correctly. However the prediction by 'multi-lingual model' misses the preposition 'in', which is present between 'Naraingarh' and 'Ambala'. The same sentence can be written in *Bengali* as follows:

- **Bengali Transliteration:** Ambālāra nārāẏanagaṛēra gāddali grāmēra kāchē ēkaṭi ṭrākēra sāthē ṭrēnēra mukhōmukhi saṅgharṣē tinajana yubaka prāṇa hārāẏa.

The phrase 'in Ambala' is represented by a single word 'Ambālāra', in *Bengali*. This difference in syntax between languages, makes the 'multi-lingual' system miss the word 'in' thus degrading the performance of the system.

The best improvement in F1-score is observed for the arguments *Reason* and *After-effects* for the *English* language. This is because these two arguments have least support in the dataset, and thus multi-lingual training helps by mitigating the scarcity in training examples. The same phenomenon can also be observed for *Reason* argument which has a low support in *Hindi* dataset. Thus through our analysis we can conclude that, 'multi-lingual' training can help in improving the performance of the system for low support classes. However, it can also cause confusion and deteriorate the performance for high support classes.

6 Conclusion

In this paper we create a dataset for argument extraction for disaster domain, for three languages *Hindi*, *Bengali* and *English*. We then build a deep learning model for extraction of these argument in each language separately. Since the data is limited in size, we build another model that leverages data from all the languages. To make use of different language datasets, we first bring the word embeddings of all the three languages to the same vector space. We also use separate language layers to accommodate divergence in syntax of the languages. Through our experiments we show that training in shared vector space by using 'multi-lingual' system helps in improving the performance of low support arguments. We also show that the for high support arguments, the syntactic difference in language can sometimes overcome the benefit of 'multi-lingual' training and cost in performance of our proposed 'multi-lingual' system.

In future we would like to explore how to handle these syntactic differences so that the performance can be further improved. It would also be interesting to explore the range of languages that can be trained successfully in a multi-lingual setting.

7 Acknowledgement

The research reported in this paper is an outcome of the project titled "A Platform for Cross-lingual and Multi-lingual Event Monitoring in Indian Languages", supported by IMPRINT-1, MHRD, Govt. of India, and MeiTY, Govt. of India.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 976–986. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176.

Zheng Chen and Heng Ji. 2009. Language specific issue and feature exploration in chinese event extraction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 209–212.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1.

Xiaocheng Feng, Bing Qin, and Ting Liu. 2018. A language-independent neural network for event detection. *Science China Information Sciences*, 61(9):092106.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1127–1136. Association for Computational Linguistics.

Libin Hou, Peifeng Li, Qiaoming Zhu, and Yuan Cao. 2012. Event argument extraction based on crf. In *Workshop on Chinese Lexical Semantics*, pages 32–39. Springer.

Ruihong Huang and Ellen Riloff. 2011. Peeling back the layers: detecting event role fillers in secondary contexts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1137–1147. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. *Proceedings of ACL-08: HLT*, pages 254–262.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Peifeng Li, Qiaoming Zhu, and Guodong Zhou. 2013. Joint modeling of argument identification and role determination in chinese event extraction with discourse-level information. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 799–809.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018. Event detection via gated multilingual attention mechanism. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 835–844. Association for Computational Linguistics.

Teruko Mitamura, Zhengzhong Liu, and Eduard H Hovy. 2015. Overview of tac kbp 2015 event nugget track. In *TAC*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 365–371.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 886–891.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective information extraction with semantic affinity patterns and relevant regions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 717–727.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 151–160. Association for Computational Linguistics.

Fabio Petroni, Natraj Raman, Tim Nugent, Armineh Nourbakhsh, Žarko Panić, Sameena Shah, and Jochen L Leidner. 2018. An extensible event extraction system with cross-media event resolution. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 626–635. ACM.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yan-yan Zhao, Bing Qin, Wan-xiang Che, and Ting Liu. 2008. Research on chinese event extraction. *Journal of Chinese Information Processing*, 22(1):3–8.

Development of POS tagger for English-Bengali Code-Mixed data

Tathagata Raha[1], Sainik Kumar Mahata[2], Dipankar Das[3], Sivaji Bandyopadhyay[4]
[1]IIIT, Hyderabad
[234]Jadavpur University, Kolkata
[1]tathagata.raha@research.iiit.ac.in, [2]sainik.mahata@gmail.com,
[3]dipankar.dipnil2005@gmail.com, [4]sivaji_cse_ju@yahoo.com

Abstract

Code-mixed texts are widespread nowadays due to the advent of social media. Since these texts combine two languages to formulate a sentence, it gives rise to various research problems related to Natural Language Processing. In this paper, we try to excavate one such problem, namely, Parts of Speech tagging of code-mixed texts. We have built a system that can POS tag English-Bengali code-mixed data where the Bengali words were written in Roman script. Our approach initially involves the collection and cleaning of English-Bengali code-mixed tweets. These tweets were used as a development dataset for building our system. The proposed system is a modular approach that starts by tagging individual tokens with their respective languages and then passes them to different POS taggers, designed for different languages (English and Bengali, in our case). Tags given by the two systems are later joined together and the final result is then mapped to a universal POS tag set. Our system was checked using 100 manually POS tagged code-mixed sentences and it returned an accuracy of 75.29%.

1 Introduction

A **P**arts-**o**f-**S**peech (POS) Tagger is a piece of software that reads the text in some language and assigns parts of speech tags, such as noun, verb, adjective, etc., to each word/token. POS Tags are useful for building parse trees, which may be used to build textbfNamed **E**ntity **R**ecognizers (NER) or Dependency Parsers. POS Tagging is also useful for building lemmatizers, which are used to reduce a word to its root form. POS taggers for widely spoken languages have been developed in abundance. But such resources are very scarce for low resourced languages.

On the other hand, code-mixing is simply a mix of two or more languages in communication. Due to the emergence of social media, a lavish amount of digital code-mixed data is generated. This is because people nowadays are very comfortable with multilingualism. This phenomenon has produced a section of researchers, who contemplate code-mixed texts as being a new language.

As mentioned earlier, since POS tagging systems for low resourced languages are hard to come by, developing one that will cater to code-mixed text is trivial. POS tagging systems, if developed for Code-Mixed data, can lead to deciphering many complex **N**atural **L**anguage **P**rocessing (NLP) tasks and hence, we attempt to develop the same in this reported work. We try to focus on creating a POS tagger for English-Bengali code-mixed data, as languages such as Bengali are morphologically rich in nature.

Our method includes scraping of code-mixed English-Bengali tweets on Twitter and cleaning them. The Bengali words in these tweets were in Roman script. These cleaned tweets were used as a development dataset for building our system. Our system starts with tagging individual tokens of a tweet with their respective languages, either English, Bengali or Unknown. This step will give rise to segments/sub-sequences of the tweet, written in the same language. It is to be noted that tokens tagged as Unknown were discarded. The segments will then be passed to two POS taggers, one designed for English and the other designed for Bengali. The output from the POS taggers will then be joined together to get the final POS tagged, code-mixed tweet. Since the POS tagging modules of English and Bengali use different tag sets, we further map the tags to a manually defined universal POS tag set. This step produces a final POS tagged tweet with uniform tags. The architecture of the proposed model is shown in Figure 1.

The remainder of the paper is organized as follows. Section 2 documents a brief state-of-art on

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 143–149
Hyderabad, India, December 2019.

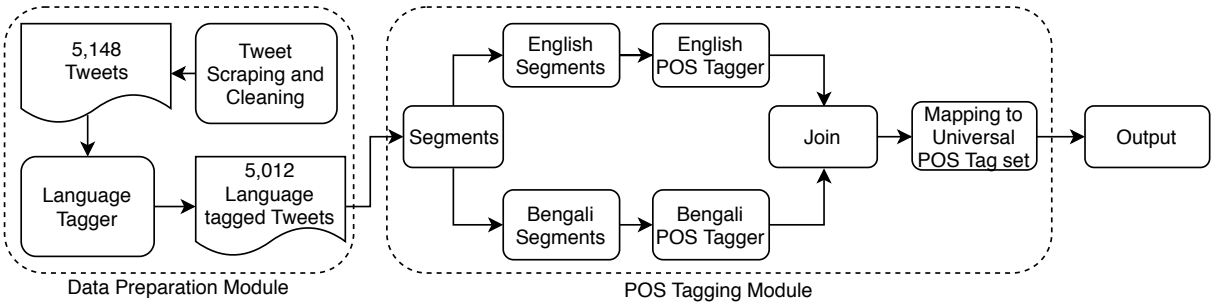


Figure 1: Architecture of the proposed model.

this domain. Section 3 defines the data preparation steps. Section 4 defines the pipeline which helps us in POS tagging the code-mixed tweets. This will be followed by the results in Section 5 and concluding remarks in Section 6.

2 Related Work

In the past few years, a lot of significant work has been done in the field of Parts of Speech tagging. The first significant POS tagger came in the early Nineties which was a rule-based tagger (Karlsson et al., 2011). One of the English rule-based taggers had an accuracy of 99.5% (Samuelsson and Voutilainen, 1997). POS taggers based on statistical approaches were also used during this time, which was based on statistical models like bi-gram,tri-gram and Markov Models (DeRose, 1988; Cutting et al., 1992; Dermatas and Kokkinakis, 1995; Meteer et al., 1991; Merialdo, 1994). Subsequently, POS tagger based on both statistical methods and a rule-based approach was proposed by Brill (1992).

Use of Conditional Random Fields for the development of POS taggers was proposed by Lafferty et al. (2001), Shrivastav et al. (2006) and Sha and Pereira (2003). Nakamura et al. (1990) used neural networks for POS tagging for the first time.

POS taggers for the Bengali language was also built by Seddiqui et al. (2003). This POStagger was built on the analysis of the Bengali morphemes. Other works have been done in Bengali POS tagging by Hasan et al. (2007) and Dandapat et al. (2007) which were rule-based and semi-supervised.

Pimpale and Patel (2016) attempted to tag code-mixed data using Stanford POS tagger. He trained the POS tagger on constrained data of Hindi, Bengali, and Telugu, mixed with English. They garnered accuracy figures of 71%. Similarly, Sarkar (2016) used the HMM model on constrained code-mixed data and achieved an accuracy figure of 75.60%.

Pipeline architecture for POS tagging of code-mixed data was first used by Barman et al. (2016). The training data was very low in their case and the LID (language identification) and transliteration models used were based on Support Vector Machines (SVM) and manual transliteration. Our approach also used pipeline architecture similar to theirs, but our model does not require any annotated data to train the system. Also, the LID and transliteration modules, in our case, have been fully trained with much larger data, using Deep Learning architecture.

3 Data Preparation

We decided to use a development dataset for building our system. It is to be noted that this data was used to build the proposed system and not to train it. Since code-mixed data consisting of English and Bengali language are difficult to find, we decided to scrape such data from Twitter. The collected tweets contained multiple degrees of noise and hence, it needed to be cleaned before using it to develop our future systems. After cleaning the tweets, they were subjected to a Language Tagger module that tagged every token of the tweet with their corresponding language (English, Bengali, and Unknown, in this case).

3.1 Tweet Scraping and Cleaning

Initially, we had to assemble the development data, consisting of English-Bengali code-mixed data, that will be used to build the POS tagger model. For this, we scraped tweets from Twitter, as it is a social media handle with a huge repository of such data. Our tweet scraper module used

the Twint module[1], a python package that helps to scrape tweets. The program was fed with a list of Bengali (Romanized) keywords that will be used to scrape the tweets. Later, the Twint object iterates the keywords and recovers tweets corresponding to the same keywords.

Using this method, 5,148 code-mixed tweets containing English and Bengali (Romanized) words were collected. The collected tweets were noisy and hence we needed to clean it beforehand to proceed. The cleaning module was a manifold approach that involved cleaning links, smileys, Emojis, Hashtags, and Mentions (Usernames).

3.2 Language Tagging and Segmentation

We observed that there is no end-to-end POS tagger available that can jointly tag English and Bengali tokens. Thus we decided to segment the cleaned tweets, into Bengali and English. This was done so that tokens in different language segments can be tagged with their respective POS tags, separately.

For segmenting the tweets, the words needed to be tagged with their corresponding language. To develop such a **L**anguage **T**agging (LT) model, we collected 11,060 Romanized words of Bengali and 7,223 words of English. We developed a binary classification model that takes as input, the tokens of a tweet (in character embedding) and outputs the language of the word to either English or Bengali. Tokens (in character embedding) were fed to a stacked LSTM of size 2. The output vectors from the LSTM cells were then fed to a fully connected layer, which then mapped the words to its specific language. For the given model, Activation was kept as *Sigmoid*, Optimizer used was *Adam* and Loss used was *Binary Crossentropy*. Batch Size was kept at 30. The program was executed for 30 epochs and the model was validated using a validation split of 0.2.

The architecture of the language tagging module is shown in Figure 2. The model returned a validation accuracy of 91%. It is to be noted that, characters apart from alphabets and numbers were tagged as Unknown'. Tweets with no language tag and only unknown tags were discarded. An example of language tagging is shown in Table 1. Statistics of the tweets after cleaning and language tagging are shown in Table 2.

After the language tagging is done, a segmentation module partitions the code-mixed input into segments concerning its language tags. In our case, segments are sub-sequences of the instance, written in the same language. An example of segmentation is shown below, where strings in brackets denote segments;

1. (Movie)$_{\text{En}}$ (ta bhalo chilo)$_{\text{Bn}}$ (but mid point)$_{\text{En}}$ (e amar khub)$_{\text{Bn}}$ (boring)$_{\text{En}}$ (lagte shuru korlo)$_{\text{Bn}}$.
2. (I had to go)$_{\text{En}}$ (karon o khub)$_{\text{Bn}}$ (urgently)$_{\text{En}}$ (daklo amaye)$_{\text{Bn}}$.

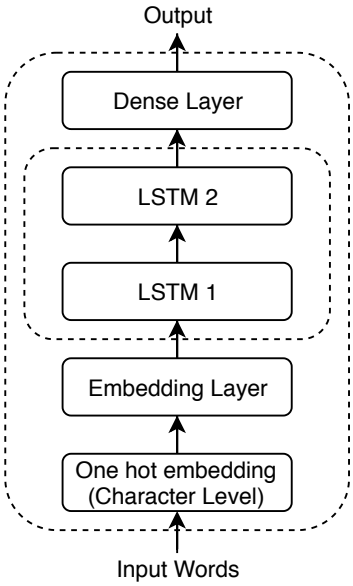


Figure 2: Language Tagging Module.

I loved the golpo and khabar ta khub nice chilo .
I\en loved\en the\en golpo\bn and\en khabar\bn ta\bn khub\bn nice\en chilo\bn .\un

Table 1: Example of Language Tagging.

3.3 Language Switch Analysis

Language tagged tweets were then analyzed to examine switching patterns. For this, the tweets were tokenized and a list of bigrams was extracted. Since the tokens of the tweets are tagged with their specific language, we could find out the count of bigrams with respect to EN-EN (both tokens of biagram are in English), BN-BN (both tokens of biagram are in Bengali), EN-BN (fist token of biagram is in English and second in Bengali) and BN-EN (fist token of biagram is in English and second in Bengali).

4 Parts of Speech Tagging

After the data preparation step, the language tagged segments are passed to the corresponding

[1]https://pypi.org/project/twint/

Particulars	Number
No. of tweets before LT	5,148
No. of tweets after LT	5,012
No. of tokens before LT	1,44,17
No. of tokens after LT	1,41,47
No. of tweets with no language tag	136

Table 2: Statistics of tweets after cleaning and Language Tagging.

Switch	Count	Freq >500	Freq >1000
EN-BN	17,758	199	88
BN-EN	17,562	166	53
EN-EN	43,859	539	203
BN-BN	16,535	98	39

Table 3: Language Switch Analysis.

language POS tagger for the final tagging. Two different POS tagging systems were used for English and Bengali. For POS tagging the English Segments we used the Stanford POS tagger[2] and the output was recorded.

For the Bengali segments, we used a tagger developed by Das et al. (2014). They trained the tagger on 10,000 Bengali (Devanagari) POS tagged sentences and tested it on 2,000 Bengali (Devanagari) sentences. Their model returned 92% accuracy. To use their model, we had to transliterate the Bengali segments into its corresponding Devanagari script. The model developed to do the same is described in Section 4.1.

4.1 Bengali Transliteration

To develop the transliteration system, we initially collected 22,781 Romanized Bengali words and manually transliterated them to its Devanagari counterpart. We developed a Sequence-to-Sequence model that takes as input the Romanized Bengali words and outputs the Bengali words in the Devanagari script. The embedding used in this model was at the character level.

The model consists of two parts: an Encoder and Decoder. The encoder takes as input, Romanized Bengali characters, creates one-hot vectors of the same and passes this to the Embedding layer. The output of the embedding layer is given to a stacked LSTM cell, which produces a context vector of the input word. The Decoder module takes as input the Bengali characters in Devanagari script, creates a one-hot vector of the same and passes it to an embedding layer. The output of the embedding layer is given to a stacked LSTM cell which is initialized with the state of the encoder module. The stacked LSTM cell then produces Bengali characters (in Devanagari script) as output, with an offset of a one-time step. The activation of the model was selected as *Softmax*, Optimizer used was *Adam* and Loss used was *Sparse Categorical Crossentropy*. Batch Size was kept at 1024. The program was executed for 50 epochs and the model was validated using a validation split of 0.1.

The validation accuracy of the model was recorded as 87%. The architecture of the model is shown in Figure 3.

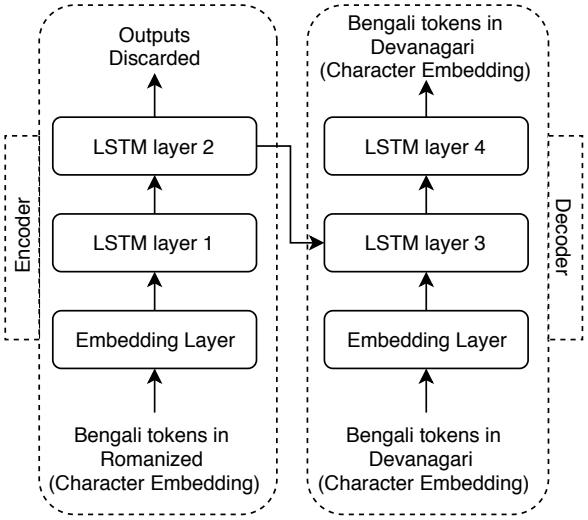


Figure 3: Back transliteration model

The transliterated segments are then fed to the Bengali POS tagger and the corresponding outputs are recorded.

After POS tagging both the English and Bengali segments, the results are joined together to get a POS tagged code-mixed tweet.

4.2 Mapping to Universal POS Tag Set

The final POS tagged code-mixed tweets need to be generalized to a universal system because the POS tags of the Bengali and English POS taggers are different. This is because English and Bengali POS taggers have different grammar and thus use different POS tag sets. To simplify this situation, we use a universal POS tag set that comprises the tags as showed in Table 4. The table shows the universal tags in bold and italics while the other texts define the universal tag.

For mapping the English POS tags to this universal POS tag set we use `map_tag` which is an inbuilt tool of NLTK. It maps the English tags to these tags based on some pre-defined rules.

The mapping of the Bengali POS tags (Stanford

[2]https://nlp.stanford.edu/software/tagger.shtml

POS	Univ.Tag	POS	Univ. Tag
Adjective	***ADJ***	Adposition	***ADP***
Determiner	***DET***	Noun	***NOUN***
Pronoun	***PRON***	Verb	***VERB***
Adverb	***ADV***	Conjunction	***CONJ***
Numeral	***NUM***	Particle	***PRT***
Punctuation	***SYM***	Other	***X***
Demonstrative	***DEM***	Intensifier	***INTF***
Reduplicative		RDP	

Table 4: Universal tag set, where text in bold and italics denote the tag and the text above define the tags

POS tags) to the universal POS tag set is shown in Table 5. Here, text in bold and italics denotes the universal tag, while the other defines the Stanford POS tags.

Syst. Tag	Univ. Tag	Syst. Tag	Univ. Tag
NN	***NOUN***	VM	***VERB***
NNP		VAUX	
INTJ		JJ	***ADJ***
PRP	***PRON***	QF	
WQ		RB	***ADV***
DEM	***DEM***	NEG	
PSP	***ADP***	RP	***PRT***
CC	***CONJ***	INTF	***INTF***
QC	***NUM***	RDP	***RDP***
SYM	***SYM***	UN	***UN***
DET	***DET***	Other	***X***

Table 5: Mapping of Bengali POS tags to the universal tagset. Text in bold and italics denotes the universal tag, while the other defines the Stanford POS tags

Finally, the POS tagged segments (mapped to the universal POS tagset) are recorded as the final output.

5 Results

Since there is no automated evaluation metric present to assess the quality of POS tagging a code-mixed sentence, we hired a linguist who was proficient in both Bengali and English. The linguist was asked to prepare a test data comprising of 100 English-Bengali code-mixed sentences. Further, the linguist was asked to POS tag the tokens, based on the universal POS tagset, separately. The linguist was told to look into the context of the sentence while tagging the tokens. This approach was used to properly

- tag ambiguous words, such as 'to', which occurs in both English and Bengali.
- tag words in the switching point.

The same test data was tagged using our system as well. To calculate the agreement between the manual annotation and system annotation, we used Krippendorff's Alpha (Krippendorff, 2011), and the metrics and the confusion are shown in Table 6

POS Tag	Man. Tag	Syst. Tag	Diff. & Conf.
NOUN	522	538	16 ADJ VERB
VERB	286	259	27 NOUN PRON
ADJ	169	141	28 NOUN VERB
PRON	104	118	14 ADJ ADV
ADV	93	63	30 VERB ADV
SYM	59	60	1 NUM
CONJ	58	49	9 NOUN VERB
DET	54	53	1 VERB
ADP	54	49	5 PRT
PRT	21	18	3 ADJ
DEM	10	11	1 NOUN
NUM	9	9	0
INTF	3	6	3 VERB
RDP	1	1	0
UN	0	606	606
K's \alpha (Interval)	0.7522		

Table 6: Agreement Analysis between manual tagged and system tagged POS tags

Inter-system annotation agreement scores described in Table 6 evaluates the overall system. To dive deeper, we evaluated every sentence of the test data. This was done using two methods.

Method 1:
For a code-mixed sentence, the POS tag of every token in the same manually annotated sentence as compared to the POS tag of every token in the same system annotated sentence. score_A was calculated as

$$\text{score}_\text{A} = \frac{\text{\# matched POS tags with manual tagged sentence}}{\text{\# tokens in the manually annotated sentence}}$$

Method 2: POS tagging of tokens that lie in the language switching point,i.e., $word_{\text{English}} \leftrightarrow word_{\text{Bengali}}$, is of utmost importance as the context of the two words may change. As a result, POS tags may also differ. In this context, score_B was calculated by multiplying 0.25 to score_A and taking the absolute value of its log value, if POS tags (for the language switching point) in the manually annotated sentence and the system annotated sentence, match. The multiplying factor was kept at 0.25 as there can be four bigrams, i.e., EN-EN, BN-BN, EN-BN, and BN-EN.

If there is more than one switching point and

the POS tags match, the multiplying factor was repeated for the number of switching. So, if there are two switching points, and the POS tags match, score_A will be multiplied by 0.25 and 0.25 to get score_B.

$$\text{score}_\text{B} = |\log(\text{score}_\text{A} * (0.25)^\text{n})|$$

, where n denotes the number of language switching points present and the trailing * denote that the formula holds true if certain conditions are met.

With the help of the above methods, Score_A and Score_B were calculated for every sentence and finally, the average for the whole test data was calculated. With method 1, our algorithm garnered accuracy of **72.72%** and with method 2, the accuracy increased to **75.29%**.

6 Conclusion

In this work, we have devised a modular system that can POS tag English-Bengali code-mixed sentences. The system uses sub-modules to perform the same. Owing to the fact, that the sub-modules can be trained for any given language, the proposed approach can be used to tag a variety of code-mixed data involving any two language pairs.

The system can be enhanced further if the sub-modules can be trained using more annotated data. E.g., if the POS tagger for the Bengali language could have been trained using more data, the problem of tagging untrained tokens with 'UN' tags could have been solved. Also, the problem of wrongly tagging tokens, e.g., tagging NOUN as ADJ, VERB and tagging PRON as NOUN, VERB, etc., could have been solved. This would have made the Bengali POS tagging module more robust. The same applies to the transliteration module as well.

In the future, we would like to develop an end-to-end system, so that the errors of one sub-module do not propagate to the other sub-modules.

References

Utsab Barman, Joachim Wagner, and Jennifer Foster. 2016. Part-of-speech tagging of code-mixed social media content: Pipeline, stacking and joint modelling. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 30–39.

Eric Brill. 1992. A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*, pages 152–155. Association for Computational Linguistics.

Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Third Conference on Applied Natural Language Processing*, pages 133–140.

Sandipan Dandapat, Sudeshna Sarkar, and Anupam Basu. 2007. Automatic part-of-speech tagging for bengali: An approach for morphologically rich languages in a poor resource scenario. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 221–224. Association for Computational Linguistics.

A. Das, U. Garain, and A. Senapati. 2014. Automatic detection of subject/object drops in bengali. In *2014 International Conference on Asian Language Processing (IALP)*, pages 91–94.

Evangelos Dermatas and George Kokkinakis. 1995. Automatic stochastic tagging of natural language texts. *Computational Linguistics*, 21(2):137–163.

Steven J DeRose. 1988. Grammatical category disambiguation by statistical optimization. *Computational linguistics*, 14(1):31–39.

Fahim Muhammad Hasan, Naushad UzZaman, and Mumit Khan. 2007. Comparison of different pos tagging techniques (n-gram, hmm and brills tagger) for bangla. In *Advances and innovations in systems, computing sciences and software engineering*, pages 121–126. Springer.

Fred Karlsson, Atro Voutilainen, Juha Heikkilae, and Arto Anttila. 2011. *Constraint Grammar: a language-independent system for parsing unrestricted text*, volume 4. Walter de Gruyter.

Klaus Krippendorff. 2011. Computing krippendorff's alpha-reliability.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Bernard Merialdo. 1994. Tagging english text with a probabilistic model. *Computational linguistics*, 20(2):155–171.

Marie Meteer, Richard M Schwartz, and Ralph M Weischedel. 1991. Post: Using probabilities in language processing. In *IJCAI*, pages 960–965.

Masami Nakamura, Katsuteru Maruyama, Takeshi Kawabata, and Kiyohiro Shikano. 1990. Neural network approach to word category prediction for english texts. In *Proceedings of the 13th conference on Computational linguistics-Volume 3*, pages 213–218. Association for Computational Linguistics.

Prakash B Pimpale and Raj Nath Patel. 2016. Experiments with pos tagging code-mixed indian social media text. *arXiv preprint arXiv:1610.09799*.

Christer Samuelsson and Atro Voutilainen. 1997. Comparing a linguistic and a stochastic tagger. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 246–253. Association for Computational Linguistics.

Kamal Sarkar. 2016. Part-of-speech tagging for code-mixed indian social media text at icon 2015. *arXiv preprint arXiv:1601.01195*.

Md Hanif Seddiqui, AKMS Rana, Abdullah Al Mahmud, and Taufique Sayeed. 2003. Parts of speech tagging using morphological analysis in bangla. In *Proceeding of the 6th International Conference on Computer and Information Technology (ICCIT)*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics.

M Shrivastav, R Melz, Smriti Singh, K Gupta, and P Bhattacharyya. 2006. Conditional random field based pos tagger for hindi. *Proceedings of the MSPIL*, pages 63–68.

Towards Handling Verb Phrase Ellipsis in English-Hindi Machine Translation

Niyati Bafna
Ashoka University
niyatisanjay.bafna_ug20@ashoka.edu.in

Dipti Misra Sharma
Indian Institute of Technology, Hyderabad
dipti@iiit.ac.in

Abstract

English-Hindi machine translation systems have difficulty interpreting verb phrase ellipsis (VPE) in English, and commit errors in translating sentences with VPE. We present a solution and theoretical backing for the treatment of English VPE, with the specific scope of enabling English-Hindi MT, based on an understanding of the syntactical phenomenon of verb-stranding verb phrase ellipsis in Hindi (VVPE). We implement a rule-based system to perform the following sub-tasks: 1) Verb ellipsis identification in the English source sentence, 2) Elided verb phrase head identification 3) Identification of verb segment which needs to be induced at the site of ellipsis 4) Modify input sentence; i.e. resolving VPE and inducing the required verb segment. This system is tested in two parts. It obtains 94.83 percent precision and 83.04 percent recall on subtask (1), tested on 3900 sentences from the BNC corpus (Leech, 1992). This is competitive with state-of-the-art results. We measure accuracy of subtasks (2) and (3) together, and obtain a 91 percent accuracy on 200 sentences taken from the WSJ corpus(Paul and Baker, 1992). Finally, in order to indicate the relevance of ellipsis handling to MT, we carried out a manual analysis of the MT outputs of 100 sentences after passing it through our system. We set up a basic metric (1-5) for this evaluation, where 5 indicates drastic improvement, and obtained an average of 3.55.

1 Introduction

Verb phrase ellipsis is a particularly frequent form of ellipsis, both in speech and in text. English VPE is the elimination of a non-finite verb phrase, introduced by an auxiliary or the particle 'to'(Kenyon-Dean et al., 2016).

We observe that state-of-the-art MT systems often cannot correctly interpret and translate sentences with VPE. For example, (elliptical phrase and site of ellipsis are in bold and italics respectively):

> Ram **cooked the food** quickly, but Shyam *did not* →
>
> *Ram ne jaldi se
> Ram-ERG - quickly-ABL -
> khana banaya, lekin Shyam
> food-OBJ cook-PT, but Shyam-ERG
> ne nahi diya.
> - not give-PT
>
> (created example)

In our initial analysis, we find that Google Translate could translate only 6/50 VPE sentences, taken from the WSJ corpus, correctly from English to Hindi. This motivated us to identify and resolve VPE in English and give this modified sentence to the machine translation system to check whether it improves the quality of translation. We used our MT system Anusaaraka for this purpose.[1]

We present a rule-based approach that performs three sub-tasks in order to solve this problem: 1) ellipsis identification 2) identification of the antecedent head verb 3) addition of necessary auxiliaries and/or complements to the head verb. Finally, it modifies the input sentence by inducing the identified verb segment at the site of ellipsis. We tailor this algorithm for the purpose of English-Hindi MT, and therefore provide a special treatment to compound verbs (including phrasal verbs),

[1] https://ltrc.iiit.ac.in/Anusaaraka/anu_home.html

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 150–159
Hyderabad, India, December 2019.

serial verbs, and verb complements in English.

Our solution is based on an understanding of verb phrase ellipsis in Hindi. Hindi does not exhibit VPE in the same manner as English; however, it exhibits a phenomenon called verb-stranding verb phrase ellipsis (VVPE) (Manetta, 2018). This means that a verbal phrase, including objects and other arguments, may be completed elided, stranding the head verb, which then appears at the site of ellipsis.

We propose that English VPE can be transferred into Hindi VVPE in order to improve MT results on the modified sentence. For this, we claim that it is enough to identify the head verb antecedent in the English sentence, and perform an analysis to support the claim.

Finally, we want to see that MT systems indeed perform better on statements with ellipsis after our treatment. Since we want to test the performance of the MT on a very specific facet: i.e. the elliptical clause, we do not utilize standard evaluation metrics but set up and define our own scale, and perform a manual analysis of 100 sentences. This evaluation dataset is meant to be merely indicative of the benefits of ellipsis handling for MT. The results are explained below.

2 Background and Our Contribution

There have been several previous works addressing the problem of antecedent head resolution for VPE in English. Cheung et al adapt the Margin-Infused-Relaxed Algorithm (MIRA) for target detection and antecedent resolution and obtain an accuracy of 65 percent (Kenyon-Dean et al., 2016). Nielson, 2005, re-implements Daniel Hardt's VPE-RES algorithm on the Penn Treebank to obtain a highest Head Overlap success of 85.87 percent and a lower Exact Match success, about 78 percent on the Brown corpus (Nielsen, 2005) (Hardt, 1992). Liu et al, 2016 experiment with various joint modelling techniques, and obtain a recall of 83.46 percent for boundary identification (Liu et al., 2016). Earlier works include Daniel Hardt's linguistically motivated rule-based system, that eliminates impossible antecedents by looking at be-do conflicts, contained antecedents and assigned scores based on co-reference of the noun subjects and clausal relationships, such as 'as' constructions. (Hardt, 1992)

The necessity for tools to deal with VPE is widely recognized in literature, for the purposes of information extraction, finding event co-occurrence, etc (Kenyon-Dean et al., 2016). In the context of MT, a possible solution is to identify the antecedent, or the source verb phrase, and induce it at the site of ellipsis to gain a legitimate, simplified sentence. This solution functions by reiterating the antecedent at the site of ellipsis. By breaking the link between the ellipsis and the antecedent, we give the MT two independent clauses, which it can translate without error.

Indeed, the previous works listed above are aimed at identifying the antecedent verb phrase, which includes compulsorily a head verb, and optionally its arguments and adverbials. While it is important to pick up arguments and adverbials of the head verb for the purpose of comprehension, there are some problems that it introduces: namely, it must often disambiguate by context and therefore can be a great source of error, and in the context of MT, it might make the output sentence clumsy and unnatural.

Since we are looking at antecedent head resolution from a particular angle i.e. transforming the English sentence containing VPE in order to align with Hindi VVPE and therefore help the MT, our problem does not require us to find the antecedent boundary of the head verb, while we do provide an additional treatment of certain verb constructions in English according to the manner in which they would be translated into Hindi. As far as we are aware, this is the first attempt to tailor English VPE resolution in the context of English-Hindi MT.

3 Ellipsis in English

Verb phrase ellipsis in English is introduced by an auxiliary. We consider five classes of el-

lipsis depending upon the auxiliary at the site of ellipsis: 1) to_be, 2) to_have, 3) to_do, 4) modals, and 5) to_particle ellipsis. Cheung et al include a sixth class: the do-so anaphora, while acknowledging that modals and Do-X anaphora are not technically auxiliaries (Kenyon-Dean et al., 2016). We have chosen to identify ellipsis introduced by modals, however, because 1. the behaviour of the former is identical with ellipsis by true auxiliaries, 2. Likewise, we observe that it poses a problem to state-of-the-art MT English-Hindi systems. We do not identify Do-X anaphora in this system, however, since this is simply a pronominalization of the antecedent rather than eliding. These have a different treatment than VPE – for example, they may be directly pronominalized in Hindi as well, rather than transferred into VVPE. Indeed, MT systems are able to do this:

> Although Mr. Azoff won't **produce films** at first, it is possible that he could *do so* later, the sources said →
>
> haalaanki, shree azoph pahalee baar
> although, Mr. Azoff first time
> philmon ka nirmaan nahin
> film-PL-POSS - production not
> karenge, lekin yah sambhav hai ki
> do-F.M, but this possible is that
> vah baad mein aisa kar sake
> he later-POST - this do can
>
> (Taken from WSJ corpus)

There are certain constraints on the antecedent, depending upon the auxiliary at the site of ellipsis. For example,

1. Auxiliaries of the form to_be require antecedents with to_be auxiliaries.

2. All non-to_be forms of ellipsis do not have an antecedent with a to_be auxiliary, except for gerunds, which are permissible.

4 Antecedent Resolution in the Context of MT

We know that state-of-the-art machines cannot interpret VPE.(Voita et al., 2019). Translation to Hindi from English requires a prediction of the elided English verb. An elided VP consists of a head verb, optionally along with its object arguments and adverbials.

For example, it is clear that the adverbial is interpreted as part of the VP in the following sentence:

> I could **walk quickly**, at that age, **across traffic-filled roads**, but now I *cannot*. (created example)

One solution to eliminate errors due to VPE, is to identify the antecedent and induce it at the site of ellipsis entirely, as a preliminary step before translation, thus eliminating the ellipsis entirely. However, this naive approach has the following issues:

1. Making the decision of whether to import a particular adjunct is complicated, and might be governed by semantic context, and (when verbal), by emphasis.

2. Reiterating the entire verb phrase at the site of ellipsis may sound clumsy and unnatural, and make for a worse translation.

Identifying the boundaries of the elided verb phrase is a much harder task than identifying the antecedent head verb. We note, for example, the consistent drop in Exact Match accuracy in Neilson's study of Hardt's algorithm over different corpora (Nielsen, 2005) – sometimes dipping as much as 30 percent lower. The matter is not as simple as picking up the entire verb sub-tree - always importing all arguments and adjuncts is not permissible, since the correct interpretation often depends on surrounding knowledge. For example, in

> Ram would have liked to eat out with you on Sunday afternoon, but he can't.
> (created example)

There are more than one interpretations of what Ram can't do: eat out, eat out with you, or eat out with you on Sunday afternoon. A native speaker selects one of these in context. If we are seeking to eliminate ellipsis with our system, we must select one of these interpretations to paste at the site of ellipsis, or the whole verb phrase. We may also not adopt intermediate policies such as importing noun objects but not adjuncts, because this risks placing undue emphasis on certain arguments, and may result in an incorrect semantics.

The second problem with this strategy is that it may render sentences, after pasting, clumsy, unnatural or nonsensical, as illustrated, respectively, by the following example, taken from the WSJ corpus (Bos and Spenader, 2011).

> The Volokhs were afraid that they'd **end up like a friend of theirs who'd applied for a visa and waited for 10 years, having been demoted from his profession of theoretical mathematician to shipping clerk.** They *didn't* (end up like…shipping clerk)

5 Hindi VVPE

Verb-stranding verb phrase ellipsis is the phenomenon wherein a verb is stranded at a site of ellipsis, and its internal arguments are elided. Manetta establishes, by various diagnostics, that Hindi-Urdu do exhibit VVPE. For example,

1. Ram-ne Chomsky-ka naya lekh do baar paRha.
   Ram-ERG Chomsky-GEN new writing two time read-PFV.M.SG

   Ram read the new paper by Chomsky twice.

2. Raj-ne bhi paRha.
   Raj-ERG also read-PFV.M.SG

   Raj also read (the paper twice). (Manetta, 2018)

Here, 'paRha' provides access to the internal arguments i.e. the direct object and the adverb via VVPE. Manetta supports her analysis theoretically and by a survey across native speakers.

This provides us an intuition for a solution for our larger problem: that is, instead of disambiguating the antecedent boundaries to resolve English VPE as simplification for English-Hindi MT, we may provide the Hindi clause containing ellipsis only the head verb, which has access to the internal arguments of the antecedent. If we induce the head verb at the site of ellipsis in the English before translating, now, then we create a valid, syntactical Hindi sentence with all the interpretations of the original sentence intact. We illustrate with an example:

a (Original sentence) Ram would have liked to eat out with you on Sunday afternoon, but he *can't*.

b (With induced head verb) Ram would have liked to eat out with you on Sunday afternoon, but he can't eat.

c Ram ko aapke saath Sunday
Ram-ERG - you-ERG with Sunday
dopahar ko baahar khana accha
afternoon-ERG - out to-eat good
lagta tha, lekin vah nahi kha
feel would-have, but he not eat
sakta.
can-M.SG

Resolving the ellipsis in (b) we get an English sentence with 'eat' at the previous site of ellipsis, that means that Ram is incapable of eating. However, (c) in Hindi, that similarly has 'eat' at previous site of English VPE, is a perfectly acceptable translation of the original sentence, exhibiting VVPE. In (c) the stranded verb 'kha' (eat) has access to the internal arguments 'aapke saath', (with you), 'Sunday dopahar' (Sunday afternoon), etc., and therefore it carries all the interpretations of the original. This is the core idea of the system.

6 Dealing with Multi-Word Verbs in English

While we do not require to identify the boundaries of the head verb, we do need to identify all the components of the verb. This may be required in several cases, such as phrasal verbs, idioms, or serial verbs.

6.1 Compound Verbs

These can be categorized into phrasal verbs and prepositional verbs. The former is a class of verbs that consists of a head verb and a preposition, like "take over", "get around" or "sink in". Since phrasal verbs are opaque, we require to pick up its preposition to maintain the correct sense in which it appears. For example, in

She hasn‘t got over her old failures as yet, although she should.
(created example)

We want the stranded segment to be ”get over”, not simply ”get”.

Prepositional verbs are also verbs followed by a preposition e.g ”stare at”, ”care for”, but they are different from phrasal verbs in certain ways. Their meaning is derived primarily from the head verb. We can see that in the first case, ”stare at”, it would admissible to strand simply ”stare”, although in the subsequent case, the sense of ‘care’ changes without the preposition. It is necessary, therefore, to pick up the prepositional verb as a unit.

6.2 Verbal Complements

In general, when the head verb has verb complement arguments, it is not necessary to pick them up, because the stranded head verb in Hindi will have access to them. For example,

a Aditya **wants to eat** dosa, although Varun *doesn‘t*. (verb complement: to eat)

b Aditya dosa khaana chahta hai,
Aditya dosa to-eat want-PSG -,
haalanki Varun nahi chahta hai.
although Varun not wants-PSG -

(b) is a legitimate translation of (a), with the stranded verb ”wants”. However, this treatment assumes that the sequence of complements in the head verb lexically translate into a sequence of simple verbs in Hindi. This might not be the case.

a Aditya **has to go** home, but Varun *doesn‘t*.

b *Aditya ko ghar jaana padega, lekin
Aditya-ERG - home to-go has-asp, but
Varun ko nahi padega.
Varun-ERG - not have-asp.

The reason we cannot strand is because ‘padega’ (has: obligation aspect) is an auxiliary of ‘jaana’ (to go) in the Hindi sentence, not the head verb of the compound verb ”jaana padega” (has to go). The head here is ”jaana”, and therefore it must be induced in the site of ellipsis. Therefore, with the English verb ”has”, among others, we must also pick up its complements to induce at the site of ellipsis. To generalize, we require the complement cE of a head verb hE, when hE + cE results in a VV or verb-light verb complex in Hindi.

6.3 Serial Verbs

Examples of serial verbs in English are

a I’ll **go see** if she’s okay

b Why don’t you **run get** a taxi?

These can be treated as verb-complement series, as the Stanford Universal Dependency Framework treats them. Hindi also treats the verb complex as a head verb followed by complements. Taking the first example, it is acceptable to strand ”jaati” (go) from this sentence, in:

a Main jaati hu dekhne ke liye
I go-P.FSG - to-see-ERG - for
ki vah theek hai ki nahi. Vah nahi
whether she okay is or not. He not
jaayega.
will-go

7 Algorithm to Identify Ellipsis and Head Verb of Antecedent

We are using a dependency tree of the input sentence for all tasks.

7.1 Identification: Rules and Results

We assume that each word w in the input sentence that belongs to our five classes, is a site of ellipsis. Then we go through a process of elimination. We have a different set of rules for each class.

Some of the basic criteria for elimination include:

1. w is a copula (for to_be)
2. w is an auxiliary child of a verb (for to_be, to_do, to_have, modals)
3. w has a direct object noun child (for to_be, to_do, to_have, modals)
4. w is not an xcomp child (for to_particle)

We perform part 1 of our two-part testing at this stage. These rules, tested on 3900 sentences from the BNC corpus (Leech, 1992), give us a precision of **94.83 percent** and a recall of **83.04 percent**.

7.2 Antecedent Head Resolution: Algorithm

If we find an ellipsis, then we perform 2 sub-tasks:

1. Find the head verb
2. Supplement the head verb

7.2.1 Finding the Head Verb

We collect all the verbs in the input, eliminating according to the constraints on to_be/non-to_be ellipsis discussed in Section 3. We use a score-based approach, as does Hardt (Hardt, 1992).

Here are the features that we look at, for verb v as a candidate for ellipsis e:

1. Positive scores for noun subjects matching in number, negative score for noun subjects not matching in 'passivity', positive score if both noun subjects are proper, positive score for identical noun subjects.
2. Negative score if e belongs to the complement clause child of v. For example, in:
   He told me that he had passed the exam, and then he told me that John hadn't.
   "told" gets ruled out as the clause "John hadn't" is a complement child of "told".
3. Negative score if v belongs to the complement clause child of e.
4. Positive score if v has an auxiliary in the same class as e. For example:
   Sita could walk while texting, but now she can't.
5. Finally, we assign a positive score to the first verb that we obtain by backtracking up the dependency tree from e, if it is contained in our candidate verbs. This gives us the correct antecedent head several times, even when it is far away. It captures clausal relationships between antecedent verb and e, such as as...as, ...than, which Hardt also mentions in their scoring algorithm.
6. We then evaluate the scores. If there is a clash, we choose the closest verb, advantaging forward ellipsis to backward ellipsis, as the latter is widely acknowledged to be much rarer than the former.

7.2.2 Supplement Main Verb

As we said earlier, the head verb might need to be supplemented before it is 'stranded'. For example, in:

> Both banks have **been battered**, as other Arizona banks *have*, by falling real estate prices.

The above algorithm will identify "battered" as the head verb; however, we do need to supplement it with the auxiliary "been", to create a grammatical verb after inducing.

We perform three sub-tasks in supplementing the main verb v:

1. Add auxiliaries: we add any auxiliaries of v, after skipping the first if any that belongs to the same class as e. For example, we skip the auxiliary "have" in the above example.
2. Add particles: here, we check whether v is part of a phrasal verb/prepositional verb, and add the preposition(s) if so. The dependency tree marks particle dependants of the verb: however, since it doesn't always do so, we maintain and import a list of common phrasal verbs/prepositional verbs for reference.
3. Add verb complements. Similarly, we maintain and import a list of verbs of which verb complements, if any, we need to pick up, since they result in non-strandable verbs in isolation in Hindi: "let", "have". These also include verbs which may not always give a correct lexical translation in isolation, e.g. "feel", "seem", for which it is safer to also induce the complements.

The lists above can always be augmented, of course. Currently, our system does not deal with idioms, but one solution that we suggest is to maintain a list of frequently occurring idioms and induce them as a whole.

8 Results and Error Analysis for Antecedent Head Resolution

In part 2 of our testing, we tested this system on 200 sentences from the WSJ corpus and got

an accuracy of 91 percent on antecedent head resolution. Here are some errors, and their analysis:

> Mr. Dinkins also has failed to allay Jewish voters' fears about his association with the Rev. Jesse Jackson, despite the fact that few local non-Jewish politicians have **been** as vocal for Jewish causes in the past 20 years as Mr. Dinkins *has*.

Here, the algorithm identifies "failed" instead of "been". It awards "failed" for common noun subject, common auxiliary and being the first verb upon backtracking, whereas "been" is only awarded for common auxiliary.

> But Sony also says in its filing that the Warner contract "doesn't require that Guber and Peters **take** any affirmative steps to produce motion pictures; it simply rewards them when they *do* and prohibits them from producing for another entertainment company."

Here the algorithm identifies "rewards" as the source verb instead of "take". It awards "take" for noun subject number, "require" for common auxiliary, and "rewards" by backtracking. Finally, it resolves the tie by choosing "rewards" which is the closest.

> Now they **know** who you mean and you know who you mean - but no one else *does*.

The algorithm gives "mean" instead of "know". "mean" gets awarded for noun subject number, and "know" is awarded for backtracking; however, "mean" wins the tie since it is the closest.

There are errors introduced by the POS tagger; for example, in:

> A good half-hour into breakfast at the Palmer House, Mr. O'Brien looks up from his plate after Mr. Straszheim says something about people who believe interest rates are about to **nosedive** - "I'm one of them who hope they *will*, with 6 billion in debt on the books.

"nosedive" is not recognized as a verb, similar to, in another examples, "program-trade", and "move".

There are errors introduced by the parser:

> The text by Patrick O'Connor is a tough read, but the pictures make her magnetism clear and help explain why Ernest Hemmingway called Baker, "The most sensational women anybody ever **saw** – or ever *will*."

Here, the algorithm wrongly identifies "called" as the antecedent main verb, instead of "saw". The clause "or ever will" is labelled a conjugate dependent of the noun Baker, instead of a conjunct of the verb "saw". If this had been so, "saw" would have got scores for a common subject ("anybody") and being the verb obtained upon backtracking.

> "A lot of people think I will **give** away the store, but I can assure you I *will not*," he says.

The algorithm identifies "assure". This would be avoided if the dependency marked "I will not" as a complement clause of "assure" – however the dependency misses this relation. If "assure" was given a penalty on this grounds, the next highest candidate is indeed the correct one: "give".

9 Evaluation of Effect on MT outputs

We now show that inducing the head elliptical verb makes the input sentence easier for the MT system to comprehend. We perform a manual analysis of 100 sentences with ellipsis, taken from the WSJ corpus: we create a "before" and "after" translation pair for each, and compare to identify improvements. This was done by two fluent speakers of Hindi and English. We define a scale (1-5) to quantify this improvement:

5: Improvement from incoherent to perfect translation of ellipsis, and surroundings
4: The meaning is fairly clearer than it was in the original
3: The translation is as good or as bad as it

originally was
2: The meaning is fairly more obscure than it originally was
1: The sentence is rendered completely incoherent from an original good translation

We add some flags to further nuance this scale: we also mark the translations for fluidity, i.e. if the translation while rendered better is still not fluent (f) or if the translation while not making the meaning clearer is rendered more fluent (F), and for the overall meaning of the entire sentence – i.e. beyond the clauses of the antecedent and the ellipsis. These markers, however, are only for exceptional cases, since most of the translations we got, both "before" and "after" were not fluent.

The average score over 100 sentences was found to be 3.55, with 18 cases of correct non-fluent sentences, and 5 cases of incorrect sentences with especial improvement in fluency.

These are the large-scale sources of lack of improvement that we found:

1. Sentences that the MT cannot translate overall due to other complexities such as nested clauses, etc., for which its output is close to gibberish, show little to no improvement with addition of the elliptical verb. These sentences passed through a system that can handle them, sometimes Google Translate, almost always show high improvement with the addition of the verb. We had about 35 such sentences, all marked 3, sometimes marked for improvement or deterioration in fluency.

2. Most sentences of the type "as did", as in 'X **ate** apples *as did* Y' fail to show any improvement and often show a deterioration in fluency. This is because the processed sentence is rendered ungrammatical and perhaps more incomprehensible. Again, Google Translate often shows improvement on these sentences from "before" to "after".

An example of the first instance is:

(1) American Enterprise Institute scholar Norman Ornstein in the Oct. 21 TV Guide on "What TV News Doesn't **Report** About Congress – and *Should*"

The system induces the elliptical verb complex "report". Both Anusaaraka and Google output incorrect translations for the original sentence, although Google shows errors only due to the ellipsis. Therefore, it is able to improve its translation after we induce the elliptical verb:

(2) amerikee entarapraij insteetyoot ke
American Enterprise Institute-POS -
vidvaan norman orsteen ne teevee
scholar Normal Ornstin-ERG - T.V.
gaid mein "kaangres ke baare
guide in "Congress-ERG - about-in
mein kya teevee riport nahin hai -
- what T.V. report not does –
aur riport karanee chaahie":
and report does should"

(Score: 5)

Whereas Anusaarak outputs:

(3) American Enterprise sansthaan vidvaan
American Enterprise institute scholar
vastushaili Ornstein par T.V. guide
Norman Ornstein on TV guide
mein: "TV samaachaar vat congress
in "TV news what congress
about report nahi does.. aur haal
about report not does… and recently
likhta hai chahiye"
Writes-MSG - should-inf."

(Score: 3f, for deterioration in fluency)

The original gives a similarly incorrect output, though not quite as ungrammatical. This output, even after the elliptical verb has been added, conveys little to no meaning in Hindi.

Here are some micro-level sources of non-improvement:

1. When an antecedent verb is being used idiomatically, the MT system may interpret it literally when stranded in the elliptical clause, even if it catches the correct sense in the antecedent clause, possible because the former construction is more unusual.

2. In general, the VVPE construction fails if the system makes two different lexical

interpretations in the antecedent and elliptical clause. This may be for different reasons: e.g. on transitive verbs, since they appear in the elliptical verb without their objects, which is unnatural. The MT system will possibly attempt to catch an intransitive sense of the verb in such a situation. However, this is only in certain few cases of such verbs.

An example of both of the above is:

> During the takeover, Mr. Hahn said he would **put** his account **up** for review if WPP's bid were successful, but he *didn't*.

The system induces the elliptical verb complex "put up". Both Anusaaraka and Google output incorrect translations for the original sentence, although Google interprets the idiomatic meaning of "put up" correctly. Therefore, it is able to improve its translation after we induce the elliptical verb:

Anusaaraka makes error type 1 (literal interpretation of verb):

(4) Vah punarvalokan ke liye uska hisaab
He review-ERG - for his account
uthega yadi vap ke neelaam ki
lift-MSG if WPP-ERG - auction of
boli saphal the toh adhineekaran
bid successful be-PL then takeover-ERG
ke dauran, shreemaan Hahn ne
- during, Mr. Hahn-ERG -
kaha, parantu vah nahi utha tha.
said, but he not lift-PST - .

(Score: 3)

Google makes error type 2 (different lexical interpretation). *In this case*, it is minor, because the meaning of the sentence is restored from the original.

(5) adhigrahan ke dauraan, shree
takeover-ERG - during, Mr.
haahan ne kaha ki agar vah
Hahn-ERG - said that if DET
wpp kee bolee saphal rahee,
WPP-POSS - bid successful stays,
to vah sameeksha ke lie apana
then he review-ERG - for his
khaata rakh dega, lekin usane
account put give-asp, but he-ERG
nahin daala.
not put.

(Score: 4)

We note here that these figures are dependent upon how well the base translation system can translate the original. We performed the same analysis on samples from this dataset with Google Translation and got consistently better results per each batch of 10, and an average of 3.7. This indicative exercise is intended to give an idea of why targeted VPE handling for specific language pairs holds significance in bettering MT results.

10 Future Work

The concept and the system that we have introduced above, for handling ellipsis in a targeted manner to improve English-Hindi MT, are still in their nascent stages. There are three primary entry points for future work: firstly, the conceptual negotiation of the phenomenon in English and Hindi. We have decided, as we explain, only to induce the main verb. While this gives satisfactory results most of the times, it also fails in some cases: it might help, for example, to make a decision to induce object arguments in these cases. Secondly, the identification of VPE in English, and the antecedent resolution. We are already dealing with complex verbs, verbal complements etc. in a certain manner, but this treatment invites further and more rigorous work, both in terms in nuance with the treatment, and how exhaustive we are with the lists that we have drawn up, introducing, for example, treatment of idioms. Thirdly, the application of our system as over the input, before it is passed through the MT. There are certain problems that may be solved by pipelining this ellipsis handling differently into the MT system: we leave this to future investigation.

References

J. Bos and J. Spenader. An annotated corpus for the analysis of vp ellipsis. *Language Resources and Evaluation*, 45(4):463–494, 2011.

D. Hardt. An algorithm for vp ellipsis. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 9–14. Association for Computational Linguistics, 1992.

K. Kenyon-Dean, J. C. K. Cheung, and D. Precup. Verb phrase ellipsis resolution using discriminative and margin-infused algorithms. In *Proceedings of the 2016 Conference on Empirical*

Methods in Natural Language Processing, pages 1734–1743, 2016.

G. N. Leech. 100 million words of english: the british national corpus (bnc). 1992.

Z. Liu, E. G. Pellicer, and D. Gillick. Exploring the steps of verb phrase ellipsis. In *Proceedings of the Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2016)*, pages 32–40, 2016.

E. Manetta. Verb-phrase ellipsis and complex predicates in hindi-urdu. *Natural Language & Linguistic Theory*, pages 1–39, 2018.

L. A. Nielsen. *A corpus-based study of Verb Phrase Ellipsis Identification and Resolution*. PhD thesis, King's College London, 2005.

D. B. Paul and J. M. Baker. The design for the wall street journal-based csr corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics, 1992.

E. Voita, R. Sennrich, and I. Titov. When a good translation is wrong in context: Context-aware machine translation improves on deixis, ellipsis, and lexical cohesion. *arXiv preprint arXiv:1905.05979*, 2019.

A Multi-task Model for Multilingual Trigger Detection and Classification

Sovan Kumar Sahoo[1], Saumajit Saha[2], Asif Ekbal[1], Pushpak Bhattacharyya[1]
[1]Department of Computer Science and Engineering, [2]Department of Mathematics
Indian Institute of Technology Patna
{sovan.pcs17, saumajit.mtmc17, asif, pb}@iitp.ac.in

Abstract

In this paper we present a deep multi-task learning framework for multilingual event and argument trigger detection and classification. In our current work, we identify detection and classification of both event and argument triggers as related tasks and follow a multi-tasking approach to solve them simultaneously in contrast to the previous works where these tasks were solved separately or learning some of the above mentioned tasks jointly. We evaluate the proposed approach with multiple low-resource Indian languages. As there were no datasets available for the Indian languages, we have annotated disaster related news data crawled from the online news portal for different low-resource Indian languages for our experiments. Our empirical evaluation shows that multi-task model performs better than the single task model, and classification helps in trigger detection and vice-versa.

1 Introduction

Event Extraction is an important task in Natural Language Processing (NLP). An event can be an occurrence happening in certain place during a particular interval of time. In text, the word or phrase that describes an event is called event trigger. Argument of an event refers to the attributes such as the location, time of occurrence of the event, participants involved and so on. Therefore event trigger detection, event trigger classification, argument trigger detection and argument trigger classification are the four important sub-tasks of event extraction. In our current paper, we have solved all the four problems using a Multi-task architecture. Multi-task learning (MTL), which essentially means performing more than one *related* task simultaneously, has been proven to be effective for various NLP tasks in recent times (Ruder, 2017). The key idea behind MTL is that the inductive transfer of knowledge, learned for a particular task, can help to improve the performance of another task by means of parameter sharing between tasks. According to Caruana (1997), "MTL improves generalization by leveraging the domain-specific information contained in the training signals of *related* tasks". In our current work, we have identified *detection* and *classification* of both event and arguments as two *related* tasks. As both event and argument trigger detection are sequence labelling problems, we have merged those two sub-tasks into one and used a single loss function. For the same reason, we have merged event and argument trigger classification task into one task and used another loss function. Thus in our proposed architecture, even though we have two main tasks for learning shared representation, we have basically solved four sub-tasks *viz.* event trigger detection, event trigger classification, argument detection and argument classification. Our proposed architecture has two variants which are further discussed later in this paper. As we are working with low-resource languages which have data sparsity issue, we have proposed a *multi-task, multi-lingual* architecture which is trained on both Hindi and Bengali data. Due to unavailability of training data in these two languages, we have annotated disaster related news data crawled from online news portals for our experiments.

2 Related Works

Being a very important problem in NLP, Event Extraction has already been explored by the research community for a long time. Some feature based approaches have decomposed the entire event extraction task into two sub-tasks and solved them separately (Ji and Grishman, 2008; Hong et al., 2011; Liao and Grishman, 2010). But the main problem of this approach is error propagation which is dealt by Riedel and McCallum (2011a), Riedel and McCallum (2011b), Li et al. (2013), Venugopal et al. (2014) using a joint event extraction algorithm. However both of the

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 160–169
Hyderabad, India, December 2019.

above approaches have used hand-designed feature. Nguyen and Grishman (2015) propose a Convolutional Neural Network (CNN) for automatic feature extraction. Chen et al. (2015) introduce a dynamic multi-pooling CNN which uses a dynamic multi-pooling layer according to event triggers and arguments in multi-event sentences, to capture more crucial information. In another work, Nguyen and Grishman (2016) propose a skip-gram based CNN model which allows non-consecutive convolution. Ghaeini et al. (2016) propose a forward-backward Recurrent Neural Network (RNN) to detect event triggers which can be in the form of both words or phrases. Feng et al. (2018) propose a language independent neural network which uses both CNN and Bi-LSTM for Event detection. Liu et al. (2016) propose to improve the performance of event detection by using the events automatically detected from FrameNet. Though these neural based systems perform well, they still suffer from error propagation issue. To overcome this issue, Nguyen et al. (2016) propose a joint framework with bidirectional RNN. However Liu et al. (2017) observe that joint model achieves insignificant improvements on event detection task. They analyze the problem of joint models on the task of event detection, and propose to use the annotated argument information explicitly for this task. Yang and Mitchell (2016) also propose a joint model for event and entity extraction but in document level instead of sentence level in contrast to most of the previous works. In recent years Liu et al. (2018a) introduce a cross language attention model for event detection where they focus on English and Chinese. Liu et al. (2018b) propose a novel framework to jointly extract multiple event triggers and arguments. Sha et al. (2018) propose a novel dependency bridge RNN which includes syntactic dependency relationships. Dependency relationship is also used by Nguyen and Grishman (2018). They investigate a CNN based on dependency trees to perform event detection. Orr et al. (2018) present a Gated Recurrent Unit (GRU) based model that combines both temporal structure along with syntactic information through an attention mechanism. Event extraction task has also been addressed in specialized tracks dedicated in Text Analysis Conference (TAC). Event extraction in disaster domain in English language is reported in (Tanev et al., 2008; Yun, 2011; Klein et al., 2013; Dittrich and Lucas, 2014; Nugent et al., 2017; Burel et al., 2017). However, significant attempt to build event extraction system in Indian languages is lacking. In recent times, some of the works are reported in (SharmilaDevi et al., 2017; Sristy et al., 2017; Kuila and Sarkar, 2017; Singh et al., 2017). To the best of our knowledge, this is the first attempt to solve four important subtasks of event extraction *viz.* event trigger detection, event trigger classification, argument trigger detection and argument trigger classification simultaneously in a *multi-task, multi-lingual* setting.

3 Task Description and Contributions

In this paper, we propose a *multi-task*, *multi-lingual* trigger detection and classification method for Hindi and Bengali in Disaster related news data. For a given Hindi/Bengali sentence, we perform the following tasks simultaneously:

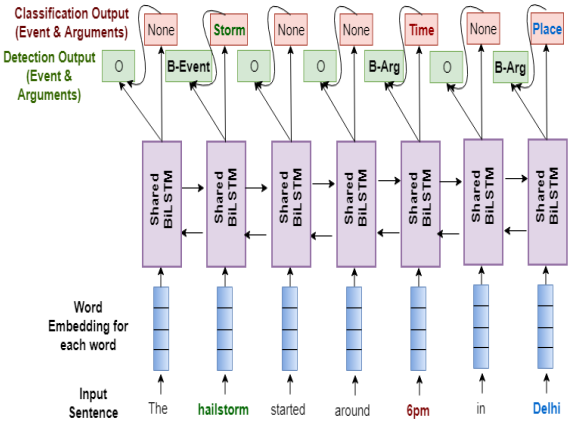
(a) Event Trigger Detection: Word or phrase that describes an event is called event trigger. Detecting event triggers is a sequence labeling task. But we formulate our current approach as a multi-class classification task as in (Chen et al., 2015; Ghaeini et al., 2016).

(b) Event Trigger Classification: Here the task is to classify each event trigger into predefined types.

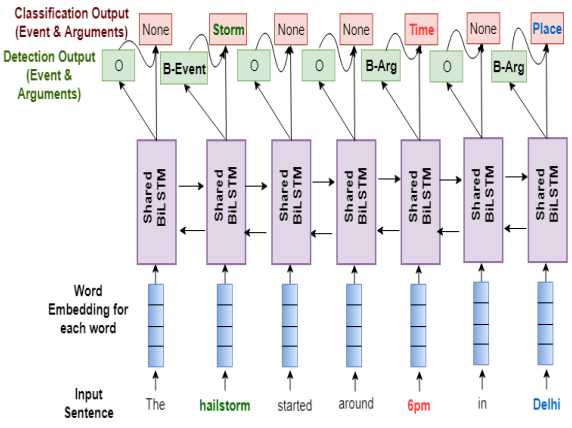
(c) Argument Detection: Arguments are entities, times or values related to an event. Here the task is to detect such trigger words or phrase.

(d) Argument Classification: Classify each argument trigger into predefined argument roles.

Argument detection is also a sequence labeling task. Like event detection, we also formulate this task as a multi-class classification problem. In most of the previous works, both event and argument detection are considered as two separate tasks. However in our current work, we combine both the tasks into a single task based on our observation. Detailed analysis of news articles reveal the fact that each type of event triggers along with its corresponding arguments follow a particular pattern in a sentence. In the first example, the sentence contains *Place* argument **दिल्ली** (**Delhi**) and *Time* argument **शाम 6 बजे** (**6pm**). Each type of argument is followed by a type specific post-position ('में' for *Place* argument and 'के' for *Time* argument). In second example the sentence contains event specific argument like *Magnitude* (**7.2**) of earthquake along with *Place* argument **इंडोनिशया** (**Indonesia**). This type of patterns are often seen in news documents. So it is intuitive to consider



(a) Proposed System 1 (MT1)



(b) Proposed System 2 (MT2)

Figure 1: Architecture of Our Proposed Models

both event and argument trigger detection as a single task. For classification also, we merge both the event trigger classification and argument trigger classification as a single task. In this way, we learn all the four above mentioned tasks simultaneously using two loss functions. We perform our experiments using both Hindi and Bengali news datasets in mono-lingual as well as multi-lingual settings. We compare our multi-task learning (MTL) results with single-task learning (STL) results for the above mentioned mono-lingual and multi-lingual settings. For most of the cases we are getting 2% to 7% performance improvement in detection task. However for classification task, we see that the performance improves for some of the classes and for the remaining classes, the model does not perform at par with the other classes. Two contributions of our paper are

- A *multi-task*, *multi-lingual* approach for event extraction in Hindi and Bengali for disaster domain. Our proposed system has two variants - (a) The *classification* output helping in *detection* (MT1). (b) The *detection* output helping in *classification* (MT2). Both the architectures are discussed in methodology section.
- Provide a benchmark setup for event extraction in Hindi language.

The following examples show that each type of event and argument trigger is followed by semantically similar kind of words in a sentence. We highlight the **event trigger** and different types of argument triggers using different colour codes for better readability.

1. **Example-1 :** दिल्ली में शाम 6 बजे के आसपास ओलावृष्टि शुरू हुई।
   Transliteration : dillee mein shaam 6 baje ke aasapaas olaavrshti shuroo huee.
   Translation : The hailstorm started around 6pm in Delhi.
2. **Example-2 :** इंडोनिशया में 7.2 की तीव्रता का भूकंप आया।
   Transliteration : indoneshiya mein 7.2 teevrata ka bhookamp aaya.
   Translation : There was a 7.2 magnitude earthquake in Indonesia.
3. **Example-3 :** शुक्रवार को अफगानिस्तान में हुई विस्फोट में 7 लोग मारे गए हैं।
   Transliteration : shukravaar ko aphagaanistaan mein huee visphot mein 7 log maare gae hain.
   Translation : 7 people have been killed in an explosion in Afghanistan on Friday.

4 Methodology

Our proposed models take sentence of the form $[w_0,w_1,...,w_n]$ as input. It produces two outputs for two main tasks namely *detection* (both event and argument) and *classification* (both event and argument). The *detection* task predicts the event or argument label (l_i) for each word (w_i) where $l_i \in$ I,O,B [1]. As we formulate *detection* as a multi-class classification task even though it being a sequence labeling task, we use *softmax* classifier at

[1] The encoding scheme is according to IOB2, where I indicates the tokens that appear within trigger, B denotes the beginning of a trigger and O denotes the outside of an event trigger. The B is used only when two events of the same type appear in consecutive sequence (Ramshaw and Marcus, 1999)

	Hindi		Bengali		Multi-Lingual	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
# of Document	681	194	799	199	1480	393
# of Sentences	12680	3077	20922	4635	33602	7712
# of Words	206882	50227	227234	45171	434116	95398
# of Event Triggers	5952	1533	7149	1602	13101	3135
# of Argument Triggers	36806	9244	44262	9058	81068	18302

Table 1: Dataset Statistics

the final layer. For *classification* task also, we use *softmax* classifier at the final layer to classify event and argument trigger into their predefined types. We employ a hard parameter sharing strategy (Caruana, 1993). We use a shared Bidirectional Long Short-Time Memory (Bi-LSTM) (Schuster and Paliwal, 1997) to capture the contextual information of each word. Figure 1a illustrates the design of first variant of our proposed architecture. Here the *classification* output of each word is concatenated with the corresponding representation resulting from the shared Bi-LSTM and fed as input to the final *detection* layer of that word. This is done with the intuition of improving the detection results with the help of classification output. For example if a word is classified as *'None'* then it has higher chance of being outside event or argument trigger boundaries. In subsequent sections, we call this architecture as MT1. Figure 1b illustrates the design of second variant of our proposed architecture. Here the *detection* output of each word is concatenated with the corresponding representation of the shared Bi-LSTM and fed as input to the final *classification* layer. This is done with the intuition of improving the classification results with the help of detection output.

4.1 Embedding

Each word of the input instance is converted to a numeric representation with the help of *fastText* (Grave et al., 2018) word embeddings having dimension 300 (d_e). The pre-trained word vectors are downloaded from *fastText* website[2]. To learn a mapping between mono-lingual word embeddings and obtain cross-lingual embeddings in order to bridge the language gap between two languages, we use the existing alignment matrices[3] which align monolingual vectors from two languages in a single vector space (Smith et al., 2017).

In order to handle *Out-of-Vocabulary* (OOV) words in the monolingual setting, we obtain their word embedding vectors from *fastText*'s `.bin` file. Separate vocabularies for OOV words are created for Hindi and Bengali respectively. We create separate `.vec` file for the two OOV vocabularies. We similarly transform these vectors of two different languages in a shared space using the existing alignment matrices[3]. It is seen that the performance has significantly improved using cross-lingual embeddings for OOV words compared to the method of using zero vectors for representing them.

5 Datasets and Experiments

5.1 Dataset

Words	Event & Argument Trigger Detection	Event & Argument Classification
इंडोनिशया	B_Arg	Place
में	O	None
7.2	B_Arg	Magnitude
की	O	None
तीव्रता	O	None
का	O	None
भूकंप	B_Event	Earthquake
आया	O	None

Table 2: Sample annotation for the sentence given in Example-2 in Task Description and Contribution Section

Since there is a lack of annotated data for our task, we create the datasets by crawling online Hindi and Bengali news articles and then annotate them following the TAC KBP[4] guidelines. For annotation, three annotators were employed. We estimate the inter-annotator agreement ratio by ask-

[2] https://fasttext.cc

[3] https://github.com/Babylonpartners/fastText_multilingual

[4] https://www.nist.gov/tac/

ing all the three annotators to annotate 5% of total documents. The multi-rater Kappa (Fleiss, 1971) agreement ratio of 0.82 and 0.85 was observed for Hindi and Bengali news documents respectively.

For both the languages, news documents are crawled from online news portal. Every sentence of news documents was pre-processed for four sub-tasks of event extraction *viz.* event trigger detection, event trigger classification, argument detection and argument classification. Table 2 presents an example of sample annotation. For detection, we use IOB2[1] format (Ramshaw and Marcus, 1999). Our proposed Hindi dataset has two types of disaster events namely natural disaster and man-made disaster which are further classified into twenty seven sub-types. Each event trigger belongs to one of the twenty seven classes, which can be found in Table 8. Every event has multiple arguments of different roles. Hindi dataset contains eleven types of arguments excluding *Type* argument type. Bengali dataset also contains eleven type of arguments excluding argument type *Intensity*. Table 5 contains all the argument types. Some of the argument types common to both Hindi and Bengali, irrespective of the event types, are *Place*, *Time*, *Casualties* and *After-effect*. Some of the arguments are specific to some particular event types. For example, *Magnitude* and *Epicentre* are event specific arguments related to *Earthquake*. Table 1 presents the dataset statistics for training and the test set of Hindi and Bengali, respectively.

5.2 Experimental Setup

Epochs	300
# LSTM units	100
Loss function for Detection	categorical_crossentropy
Loss function for Classification	categorical_crossentropy
Optimizer	Adam

Table 3: Hyper-parameter Settings

For implementing the deep learning models Python based library Keras (Chollet et al., 2015) with Tensorflow (Abadi et al., 2015) backend is used. All the models are trained for 300 epochs. Training is done using a learning rate of 0.001 and *'Adam'* optimizer is used for fast convergence. The data is fed to the neural network in batches of 32. *'Checkpoints'* are used to save the best weights of the model based on training accuracy. Table 3 shows the hyper-parameter settings used in the implementation of both the variants of our proposed model. For evaluation *precision*, *recall* and *F1-score* are used as the metrics. However in result tables (refer Table 4, Table 5, Table 6, Table 7 and Table 8) only *F1-score* is reported.

6 Results and Analysis

Table 4, Table 5 and Table 8 show the experimental results for event and argument trigger detection, argument role classification and event trigger classification respectively, where ST denotes Single task, MT1 denotes Multi-task 1, MT2 denotes Multi-task 2 and SP denotes support count. Table 4 shows that multi-task model 1 (MT1) performs well as compared to single task (ST) model for all language settings. For each language setting, performance improvement is maximum in case of *I_Event* tag. We find that it is 7.3% for Hindi, for Bengali it is 11.5% and for multi-lingual setting it shows improvement of 6.5%. Analyzing the predictions of all the variants of our system reveal that words are usually miss-classified more between the Beginning (B) and Inside (I) tag type of either event or argument instead of events getting miss-classified as argument triggers. Thus we can conclude that the system produces near correct prediction of event and argument trigger in most of the cases, only issue being that it sometimes fail to determine the correct trigger boundary. Figure 2a and Figure 2b show the confusion matrix obtained by MT1 in trigger detection and trigger classification in the multilingual setting.

6.1 Comparison With Separate Event and Argument Trigger Detection System

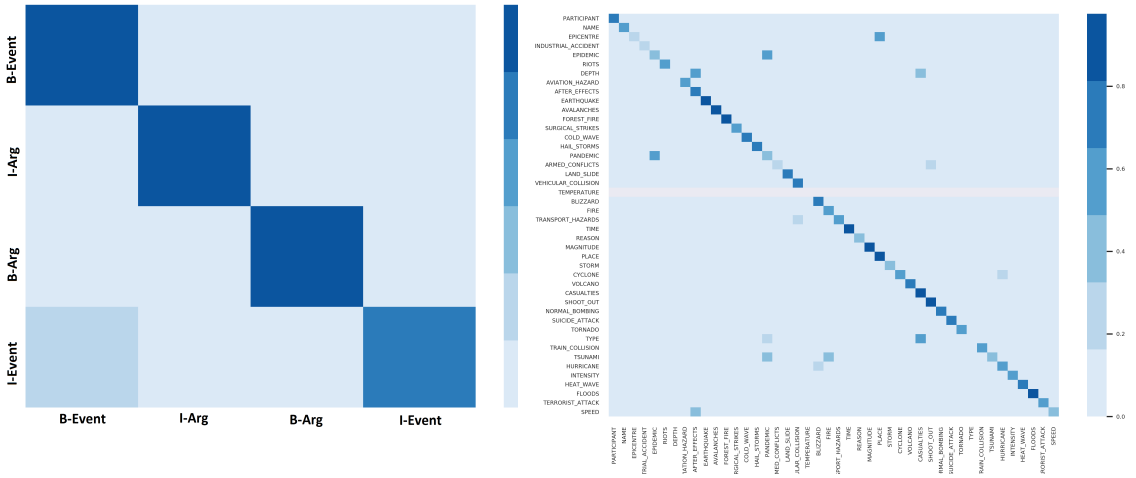
We also perform separate experiments to evaluate our proposed approach with the earlier proposed approaches of separately detecting event and argument triggers from sentences. Table 6 shows the *F1-score* achieved in event trigger detection and Table 7 shows the *F1-score* obtained in argument trigger detection for both the Hindi and Bengali datasets. The evaluation shows that there is not any significant loss in performance in simultaneous detection of event and argument triggers compared to individual trigger detection even though there is a marginal improvement in detection of the tag *I_Event* for Bengali in the argument detection model compared to the model which per-

	Hindi				Bengali				Multi-Lingual			
	ST	**MT1**	**MT2**	**SP**	**ST**	**MT1**	**MT2**	**SP**	**ST**	**MT1**	**MT2**	**SP**
B_Event	0.57	**0.59**	**0.59**	929	0.63	**0.65**	0.64	1111	**0.61**	**0.61**	**0.61**	2040
I_Event	0.41	**0.44**	0.42	594	0.52	**0.58**	0.55	491	0.46	0.48	**0.49**	1085
B_Arg	0.48	**0.5**	0.48	2476	0.57	0.57	**0.58**	2658	0.52	**0.53**	0.51	5134
I_Arg	**0.49**	**0.49**	0.46	6747	0.64	**0.65**	**0.65**	6400	0.56	**0.57**	0.55	13147

Table 4: Trigger Detection (Events and Arguments) Results

	Hindi				Bengali				Multi-Lingual			
	ST	**MT1**	**MT2**	**SP**	**ST**	**MT1**	**MT2**	**SP**	**ST**	**MT1**	**MT2**	**SP**
Participant	0.35	**0.42**	0.38	539	**0.43**	**0.43**	0.41	816	0.36	**0.41**	0.36	1355
Epicentre	**0.59**	0.46	0.29	22	**0.48**	0.27	0.46	49	**0.4**	0.2	0.35	71
After Effect	0.3	**0.35**	0.31	2828	**0.36**	**0.36**	0.35	1648	0.32	0.31	**0.33**	4476
Reason	**0.14**	0.1	0.12	354	**0.26**	0.21	0.20	280	0.16	0.16	**0.18**	634
Magnitude	0.56	0.6	**0.62**	40	**0.52**	0.51	0.44	25	0.47	**0.56**	0.54	65
Place	0.57	**0.58**	0.56	2369	**0.61**	0.59	**0.61**	1588	**0.58**	0.57	0.56	3957
Casualties	0.58	**0.59**	0.58	1969	**0.73**	**0.73**	0.72	2578	0.65	**0.66**	0.65	4547
Name	0.26	**0.32**	0.27	67	0	0	0	9	0.25	**0.3**	0.23	76
Type	-	-	-	-	0.20	0.20	**0.24**	29	0.19	0.11	**0.37**	29
Intensity	**0.54**	0.44	0.4	191	-	-	-	-	**0.45**	0.33	0.27	191
Time	0.65	**0.66**	0.63	804	0.84	**0.85**	0.84	2029	**0.79**	0.77	0.78	2833
Speed	**0.18**	0.11	0	17	0.36	0.31	**0.46**	4	0.19	**0.36**	0.27	21

Table 5: Argument Role Classification Results



(a) Trigger Detection (b) Trigger Classification

Figure 2: Confusion Matrix : MT1 in Multilingual Setting.

	Hindi	Bengali
B-Event	0.56	0.63
I-Event	0.42	0.55

Table 6: Result of Event Trigger Detection as Only Task.

	Hindi	Bengali
B-Arg	0.49	0.57
I-Arg	0.49	0.64

Table 7: Result of Argument Detection as Only Task.

	Hindi				Bengali				Multi-Lingual			
	ST	**MT1**	**MT2**	**SP**	**ST**	**MT1**	**MT2**	**SP**	**ST**	**MT1**	**MT2**	**SP**
Armed Conflicts	0.2	**0.4**	0.31	7	**0.22**	0.16	**0.22**	126	0.21	0.19	**0.24**	133
Avalanches	0.57	0.61	**0.62**	30	-	-	-	-	0.51	**0.57**	**0.57**	30
Aviation Hazard	0.35	0.43	**0.46**	43	**0.56**	0.47	0.34	34	**0.48**	0.34	0.41	77
Blizzard	0.49	**0.6**	0.51	19	0	0	0	7	0.44	0.41	**0.6**	26
Cold Wave	**0.53**	0.48	**0.53**	26	**0.50**	**0.50**	**0.50**	4	**0.52**	0.45	0.49	30
Cyclone	0.4	**0.49**	0.36	20	-	-	-	-	0.51	0.45	0.45	20
Earthquake	0.69	**0.73**	0.66	115	**0.75**	0.74	0.68	87	**0.71**	0.63	**0.71**	202
Epidemic	-	-	-	-	**0.33**	**0.33**	**0.33**	61	**0.34**	0.3	0.3	61
Fire	**0.27**	0.26	0.25	114	**0.68**	**0.68**	0.66	120	0.44	0.45	**0.48**	234
Floods	0.56	0.6	**0.7**	27	0.40	**0.67**	0.50	1	0.64	**0.77**	0.66	28
Forest Fire	**0.32**	0.31	0.29	63	-	-	-	-	**0.33**	0.3	0.24	63
Hail Storms	0.41	**0.46**	0.39	41	-	-	-	-	0.45	**0.52**	0.46	41
Heat Wave	0.39	**0.48**	0.39	66	0.33	0.24	**0.43**	9	0.36	0.37	**0.41**	75
Hurricane	0.53	**0.6**	0.38	35	-	-	-	-	**0.48**	0.47	0.45	35
Industrial Accident	**0.21**	**0.21**	0.17	113	0	**0.25**	0	3	0.17	**0.18**	0.15	116
Landslide	0.43	0.38	**0.44**	69	**0.74**	0.71	0.59	9	0.47	**0.5**	0.46	78
Normal Bombing	0.18	0.2	**0.22**	9	0.61	**0.62**	0.58	292	**0.57**	0.55	0.56	301
Pandemic	-	-	-	-	**0.26**	0.23	0.25	87	0.17	0.29	**0.32**	87
Riots	0.29	**0.38**	0.31	32	0.26	**0.31**	0.23	44	**0.28**	0.2	0.24	76
Shootout	**0.49**	**0.49**	0.44	110	**0.56**	0.54	0.52	177	0.51	**0.52**	0.5	287
Storm	0.2	0.22	**0.29**	24	**0.45**	0.42	0.42	26	**0.43**	0.32	0.34	50
Suicide Attack	0.64	0.64	**0.68**	154	0.57	**0.62**	0.56	123	**0.6**	0.59	0.58	277
Surgical Strikes	0	0	0	2	0.40	0.36	**0.44**	64	**0.41**	0.38	0.36	66
Terrorist Attack	0.61	0.61	**0.62**	95	0.32	**0.37**	0.34	147	0.47	0.48	**0.49**	242
Tornado	0.43	**0.49**	0.35	32	**0.57**	**0.4**	**0.57**	4	**0.43**	0.38	**0.43**	36
Train Collision	0.52	0.44	**0.53**	72	0	0	0	1	0.46	0.4	**0.5**	73
Transport Hazards	0.13	**0.18**	**0.18**	79	**0.49**	0.47	0.43	127	**0.4**	0.36	0.37	206
Tsunami	-	-	-	-	**0.17**	**0.17**	**0.17**	10	**0.32**	0.13	0.12	0.32
Vehicular Collision	**0.56**	0.52	0.49	93	0.43	0.45	**0.48**	39	0.44	**0.48**	0.46	132
Volcano	0.5	0.42	**0.52**	33	-	-	-	-	**0.48**	0.45	0.43	33

Table 8: Event Trigger Classification Results

forms simultaneous detection of both triggers.

6.2 Error Analysis

In the following Input Example 1, ज्वालामुखी विस्फोट (**jvaalaamukhee visphot\volcanic eruptions**) is a multi-word event trigger. The tags assigned for this trigger are *B_Event* and *I_Event* respectively. In Input Example 2, the event trigger विस्फोट (**visphot\eruptions**) is tagged as *B_Event*. For the first case, all the variants of the sys-

tem predict the event trigger correctly but for the later case, our single task detection system (ST) and multi-task system 2 (MT2) predict it as outside event and argument trigger boundary (*O*) but multi-task system 1 (MT1) predicts it as inside event trigger (*I_Event*) rather than beginning of event trigger (*B_Event*). Thus we can see that all the variants miss-classify the trigger tag with MT1 being able to produce partially correct prediction as it, at least, classifies it to be of event type. However the classification result of the said event trigger in example 2 is correctly predicted by MT1 but it is wrongly predicted by MT2. Here we can see that the classification task is helping in detection task.

1. **Input Example 1 :** अमरीका में **ज्वालामुखी विस्फोट** को लेकर रेड अलर्ट जारी ।
   Transliteration : amareeka mein **jvaalaamukhee visphot** ko lekar red alart jaaree.
   Translation : US issues red alert for **volcanic eruptions**.

2. **Input Example 2 :** उल्लेखनीय है कि बीते कुछ दिनों से माउंट अगुंग ज्वालामुखी में छोटे-छोटे **विस्फोट** हो रहे है।
   Transliteration :ullekhaneey hai ki beete kuchh dinon se maunt agung jvaalaamukhee mein chhote-chhote **visphot** ho rahe hai.
   Translation : It is notable that in the last few days, small **eruptions** in the Mount Agung Volcano.

We provide below a detailed error analysis of the results achieved in classification task (refer to Table 5 and Table 8).

1. In the classification task (refer to Table 5), error analysis reveals that the performance is affected mainly due to two cases : (a) when the Support count of a trigger type is less, (b) when each trigger mention in a sentence is long, `i.e.` it consists of numerous words. For example, *Participant, Time, Place, Casualties* and *Intensity* have better *F1-score* as the trigger mentions corresponding to these types are in the form of short phrases as well as these types have larger support count. However, roles like *After Effect* and *Reason* have comparatively lower performance as these trigger mentions appear in sentences in the form of long phrases. Even though *Magnitude* has less support count, performance is better compared to the other roles as the trigger mention is in the form of a single word comprising of a numeric figure.

In Table 8, we observe the following drawbacks which can possibly lead to erroneous output.

1. We find that performance decreases for similar types of events. For example, types like *Fire, Forest Fire and Industrial Accident* are of similar type. We see that the performance of these types is low in Hindi as all of them are present in the dataset, thereby getting miss-classified. However in Bengali dataset, we find *Fire* performs relatively better as there does not exist any sentence having event trigger of type *Forest Fire* and *Industrial Accident*.

2. In Hindi dataset, we find that type *Transport Hazard* is seen to be misclassified with type *Train Collision* and type *Vehicular Collision*, therby leading to poor performance. For Bengali dataset, there hardly exists any trigger of type *Train Collision* and event trigger of type *Vehicular Collision* exists in small number. Thus Bengali dataset performs much better for *Transport Hazard*.

7 Conclusion and Future Works

In this paper, we present a *multi-tasking, multi-lingual* architecture for simultaneous *detection* and *classification* of event and argument triggers. We have proposed two variants where in each one of them, one task is helping another related task. Our results show that related tasks can definitely share information between them. We also compare our approach with separate models which can be employed for event and argument trigger detection respectively.

Other future works include developing an end-to-end system which will consist of a *multi-tasking* system such that given a sentence as input, event and argument triggers will be extracted from it and if there exists any link between the extracted event and argument, then the output of the system will be positive and otherwise negative.

8 Acknowledgement

The work reported in this paper is supported by the project titled “A Platform for Cross-lingual and Multi-lingual Event Monitoring in Indian Languages”, sponsored by IMPRINT-1, Ministry of

Human Resource and Development, Government of India. Sovan Kumar Sahoo gratefully acknowledges "Visvesvaraya PhD Scheme for Electronics and IT", under the Ministry of Electronics and Information Technology, Government of India.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Grégoire Burel, Hassan Saif, Miriam Fernandez, and Harith Alani. 2017. On semantics and deep learning for event detection in crisis situations. *Workshop on Semantic Deep Learning (SemDeep), at ESWC 2017, 29 May 2017, Portoroz, Slovenia*.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Richard Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48. Morgan Kaufmann.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 167–176.

François Chollet et al. 2015. Keras. `https://keras.io`.

André Dittrich and Christian Lucas. 2014. Is this twitter event a disaster? *AGILE Digital Editions*.

Xiaocheng Feng, Bing Qin, and Ting Liu. 2018. A language-independent neural network for event detection. *Science China Information Sciences*, 61(9):092106.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Reza Ghaeini, Xiaoli Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 369–373.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning Word Vectors for 157 Languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, pages 3483–3487.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1127–1136. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. *Proceedings of ACL-08: HLT*, pages 254–262.

Bernhard Klein, Federico Castanedo, Inigo Elejalde, Diego López-de Ipina, and Alejandro Prada Nespral. 2013. Emergency event detection in twitter streams based on natural language processing. In *International Conference on Ubiquitous Computing and Ambient Intelligence*, pages 239–246. Springer.

Alapan Kuila and Sudeshna Sarkar. 2017. An event extraction system via neural networks. *FIRE (Working Notes)*, pages 136–139.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 73–82.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018a. Event detection via gated multilingual attention mechanism. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2134–2143.

Shulin Liu, Yubo Chen, Kang Liu, Jun Zhao, et al. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018b. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 365–371.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 886–891.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Tim Nugent, Fabio Petroni, Natraj Raman, Lucas Carstens, and Jochen L Leidner. 2017. A comparison of classification models for natural disaster and critical event detection from news. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 3750–3759. IEEE.

Walker Orr, Prasad Tadepalli, and Xiaoli Fern. 2018. Event detection with neural networks: A rigorous empirical evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 999–1004.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Sebastian Riedel and Andrew McCallum. 2011a. Fast and robust joint models for biomedical event extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–12. Association for Computational Linguistics.

Sebastian Riedel and Andrew McCallum. 2011b. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 46–50. Association for Computational Linguistics.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

V SharmilaDevi, S Kannimuthu, and G Safeeq. 2017. Kce_dalab@ eventxtract-il-fire2017: Event extraction using support vector machines. *FIRE (Working Notes)*, pages 144–146.

Jyoti Prakash Singh, Yogesh K Dwivedi, Nripendra P Rana, Abhinav Kumar, and Kawaljeet Kaur Kapoor. 2017. Event classification and location prediction from tweets during disasters. *Annals of Operations Research*, pages 1–21.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Nagesh Bhattu Sristy, N Satya Krishna, and Durvasula VLN Somayajulu. 2017. Event extraction from social media text using conditional random fields. In *FIRE (Working Notes)*, pages 140–143.

Hristo Tanev, Jakub Piskorski, and Martin Atkinson. 2008. Real-time news event extraction for global crisis monitoring. In *International Conference on Application of Natural Language to Information Systems*, pages 207–218. Springer.

Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 831–843.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. *arXiv preprint arXiv:1609.03632*.

Hong-Won Yun. 2011. Disaster events detection using twitter data. *Journal of information and communication convergence engineering*, 9(1):69–73.

Converting Sentiment Annotated Data to Emotion Annotated Data

Manasi Kulkarni[1,2]
[2]Computer Engineering and IT
Veermata Jijabai Technological Institute
Mumbai
manasi@cse.iitb.ac.in

Pushpak Bhattacharyya[1]
[1]Computer Science and Engineering,
Indian Institute of Technology Bombay
Mumbai
pb@cse.iitb.ac.in

Abstract

Existing supervised solutions for emotion classification demand large amount of emotion annotated data. Such resources may not be available for many languages. However, it is common to have sentiment annotated data available in these languages. The sentiment information (+1 or -1) is useful to segregate between positive emotions or negative emotions. In this paper, we propose an unsupervised approach for emotion recognition by taking advantage of the sentiment information. Given a sentence and its sentiment information, recognize the best possible emotion for it. For every sentence, the semantic relatedness between the words from sentence and a set of emotion-specific words is calculated using cosine similarity. An emotion vector representing the emotion score for each emotion category of Ekman's model, is created. It is further improved with the dependency relations and the best possible emotion is predicted. The results show the significant improvement in f-score values for text with sentiment information as input over our baseline as text without sentiment information. We report the weighted f-score on three different datasets with the Ekman's emotion model. This supports that by leveraging the sentiment value, better emotion annotated data can be created.

1 Introduction

An emotion or a feeling represents a state of mind for any person. Various researchers have put-forward classification of emotions into various categories such as Plutchick emotion model with 8 basic emotions (Plutchick 1980), the Ekman's Model with six basic emotions – anger, disgust, fear, happy, surprise, sadness (Ekman 1972) and so on. Users easily share their experiences, opinions, and emotions on various topics, product reviews on social platforms such as Twitter, Facebook, Whatsapp. Understanding the emotions expressed in such short posts can facilitate many important downstream applications such as an emotion-aware chatbots, analysis of user reviews, personalized recommendations, a help to psychologically ill patients, and so on. Therefore, it is important to develop the effective emotion recognition models to automatically identify emotions from such text or messages.

The task of emotion detection is typically modelled as supervised multi-class classification or multi-labelled classification task. Supervised models need very large annotated data. Such datasets may not be readily available and are costly to obtain (Jianfei Yu, 2018). In case of unavailability of annotated data, unsupervised learning approaches (A Agrawal, 2012; Milagros Fernández-Gavilanes, 2015) can be an ideal solution for the emotion recognition.

However, we assume that text annotated with sentiment information (positive, negative or neutral) is easily available. Sentiment classification predicts positive or negative sentiment polarity of a sentence whereas emotion classification labels the sentence at fine-grain level with one of the emotions, such as happy, surprise, anger, fear, disgust, sadness etc. Happy and surprise emotion are termed as the positive sentiment emotions and anger, disgust, fear, sadness as the negative sentiment emotions. For example in the sentence,

Passed an exam by two points . . . (+1)

The sentiment information provided with sentence in above example, helps to confirm that the sentence is made with positive emotion such as happiness, surprise etc rather than the negative emotions.

A close friend of mine have not contacted me long time. . . . (-1)

Sentiment value of -1 shows exclusion of positive emotions by reducing chances of emotion recognition system being confused them with positive

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 170–177
Hyderabad, India, December 2019.

emotions. The sentiment information helps to narrow down choices to one of the negative emotions for the sentence and it must be sadness, fear, anger, or disgust.

Therefore, we aim to use this sentiment information along with the sentence to create the emotion labelled dataset by recognizing emotion in unsupervised way. To be precise to create the emotion labelled resources with help of available sentiment labelled resources, as a further fine grained emotion analysis task.

In this paper, we propose an unsupervised approach based on A Agrawal (2012), with our modifications as discussed later in detail. We use the sentiment labelled data that is the sentence and the respective sentiment information as input and recognize the best possible emotion for the same. This approach uses word-embeddings to represent the words in the sentences as well as emotion-specific words. The cosine similarity measure is used to calculate the semantic relatedness between a sentence and the emotion-specific words. The vector representing score for every emotions category is calculated for each sentence and then emotion-score is modified using dependency relations of open-class words from that sentence.

The rest of the paper is organized as follows. The section 2 describes related supervised, unsupervised and hybrid approaches previously proposed in the literature. The section 3 discusses the methodology of proposed system. Section 4 briefs on the experimental set up, datasets used as well as modifications done to the dataset as required for experimentation. The results are discussed in section 5.

2 Related Work

The state of the art approaches for emotion Recognition task is supervised approach. The labeled training data is a crucial resource required for building such systems. Due to the lack of a large human annotated datasets, many emotion classification tasks have been performed on text data gathered from social media such as twitter, and the hash-tags, emojis or emoticons are used as the emotional labels for the same.

As the unsupervised approaches do not need the annotated dataset, different unsupervised approaches are also performed by researchers. A Agrawal (2012) found open-class words which they named as NAVA words that is Noun, Adjective, Verb, Adverb words. Pointwise Mutual Information (PMI) based model with syntactic dependencies is used to perform emotion recognition. Shoushan Li and Zhou (2015) created a Dependence Factor Graph (DFG) as learning model based on label dependence and context dependence. The hybrid approaches use the unsupervised approaches for feature creation, pattern extraction which are later used by supervised classification models for emotion classification. Carlos Argueta (2015) had proposed an unsupervised graph-based approach for boot-strapping the Twitter-specific emotion-bearing patterns and then used them for classification task. Li and Xu (2014) used predefined linguistic patterns to extract emotion causes and considered them as features for classification using SVM.

Jianfei Yu (2018) have used transfer learning approach for sentiment classification task and then emotion classification task. Also few researchers have contributed towards creation of emotion-aware embedding. Distant supervision and Recurrent Neural Network (RNN)-based approach is proposed for learning emotion-enriched representations.(Ameeta Agrawal, 2018)

3 Proposed System

The emotion recognition framework for unsupervised approach is as shown in Figure 1. The input sentence is pre-processed to get the open-class words from that sentence. The second step is to compute the semantic relatedness using cosine similarity between word embedding of words in sentence and emotion-specific words. The module three modifies the emotion score for every emotion from vector computed at module 2. Later, module 4 averages over emotion vectors of all words of a sentence to find resulting emotion present in that sentence.

Let S be the sentence, $S =< w_1, w_2, \cdots, w_n >$ and S_s be the respective sentiment value (+1 or -1 or 0). Let E be the set of possible emotions from selected emotion model such as $E = \{e_1, e_2, \cdot \cdot \cdot, e_m\}$. To every emotion category, we have assigned few affect bearing words which represent that emotion. Table-1 shows few affect-bearing words used for each emotion category. The aim is to predict the best possible emotion Es belongs to

E for the sentence S. Eventually, one of the emotions from the Ekman's emotion model- Anger, Disgust, Fear, Happy, Sadness and Surprise will be predicted for every sentence.

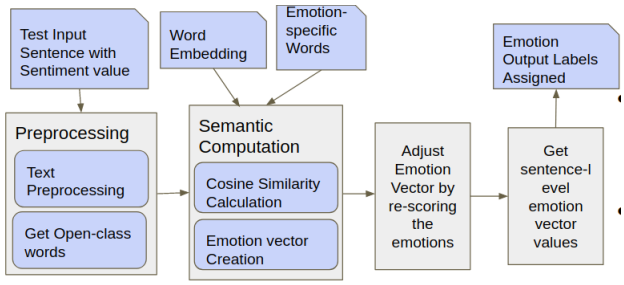


Figure 1: Overview of System

3.1 Computing Semantic Relatedness using Cosine Similarity

We have used pre-trained word embedding as they better represent co-occurrence information of words. The words in a given sentence and the emotion-specific words are represented using their respective word embedding and the semantic relatedness between them is found using cosine similarity. Let A and B be word vector representation for 2 words then:

$$sim(A, B) = \cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||}$$

3.2 Computing Vector of Scores for Emotion Categories

The emotion score vector for every open-class words $\{w_1, w_2, \cdots, w_n\}$ of a sentence is created. The length of the emotion vector is six values corresponding to six emotions of Ekman's model.

The emotion vector for a word w_i is computed by finding cosine similarity of word w_i with every emotion-specific word from each emotion category. Let there be m emotion categories and $\{EW_1, EW_2, \cdots, EW_m\}$ be sets of l emotion-specific words for each emotion e_j. Then the emotion score ES for w_i is calculated as:

$$ES(w_i, e_j) = \sum_{k=1}^{l} sim(w_i, EW_j^k) \qquad (1)$$

$$\forall j = 1 \ldots m$$

An emotion score vector EV for every word is created using a sentiment value and an emotion scores of corresponding emotions with given sentiment value. So, an emotion-score vector for word w_i is,

$$EV_{w_i} = \{ES(w_i, e_1), ES(w_i, e_2), \ldots, ES(w_i, e_m)\} \qquad (2)$$

Here, the sentiment value plays a crucial role. As we consider the Ekman's emotion model, happy and surprise emotions are emotion with positive sentiment. The emotions such as fear, disgust, angry and sadness are emotions with negative sentiment. Hence,
if *sentiment* = $+1$ then,

$$EV_{w_i} = \{ES(w_i, happy), ES(w_i, surprise), 0, 0, 0, 0\} \qquad (3)$$

if *sentiment* = -1 then,

$$EV_{w_i} = \{0, 0, ES(w_i, anger), ES(w_i, disgust), ES(w_i, fear), ES(w_i, sadness)\} \qquad (4)$$

Emotion	Emotion Words
Anger	anger, angry, annoy, irritate, frustrate
Disgust	disgust, hate, dislike, ill, sick
Fear	fear, worry, terrify, afraid, frighten
Happiness	happiness, happy, love, joy, glad
Sadness	sadness, sad, hurt, cry, bad
Surprise	surprise, amazing, astonish, wonderful, incredible

Table 1: Few affect-bearing words used

3.3 Re-scoring Scores in Emotion Vector of a Word

With the intuition that dependency relationship can contribute more towards emotion detection, we use these relations of open class words to modify the emotion vector EV of the dependent word. The Stanford's coreNLP dependency parser is used for finding dependencies between the open-class words that is noun, adjective, verb and adverb which are considered for further processing.

Let $sd(w_1, w_2)$ be a syntactic dependency relation between word w_1 as dependent and word w_2 as modifier. For example, in adjectival modifier relation *amod(life, happy)*, dependent word is *life* and modifier word is *happy*. We find these syntactic dependencies of sentence at the time of preprocessing itself and use dependency relations for the re-scoring purpose.

Let D be the dependent word from sentence S and M be the respective modifier word from sentence S. Then the emotion vector D_p of p^{th} word is modified by taking average over emotion vectors of the dependent word D_p and its modifier word M_p. This will help in strengthening every emotion score related to that word w_i of sentence S. (A Agrawal, 2012)

$$EV_{D_p} = \frac{EV_{D_p} + EV_{M_p}}{2} \quad (5)$$

3.4 Processing Emojis

With growing usage of social media, many times, text messages are accompanied with suitable emoji. Hence, emoji as input can contribute towards detecting emotion. Every emoji is being assigned CLDR short name by Unicode Common Locale Data Repository to describe that emoji. For example, *grinning face, beaming face with smiling eyes* and so on. Same procedure as mentioned in section 3.1 to 3.3 is followed for creating emotion vector M_{EV} for every emoji.

3.5 Computing Emotion Vector for Sentence

The emotion vector S_{EV} for the sentence S is calculated by taking average over emotion vectors EV_{w_i} of all words from that sentence,and emoji emotions vector M_{EV}.

3.6 Resultant Emotion Prediction

The emotion vector of text S is :

$$S_{EV} =< S_{e_1}, S_{e_2}, \ldots, S_{e_m} >$$

where the emotion vector for emoji, if present in sentence is:

$$M_{EV} =< M_{e_1}, M_{e_2}, \ldots, M_{e_m} >$$

and

$$S_{EV} = \frac{1}{n} \sum_{i=1}^{n} EV_{w_i} + M_{EV} \quad (6)$$

the best possible predicted emotion Es for S as:

$$Es = argmax(S_{e_i}) \quad (7)$$

$$\forall i = 1 \ldots m$$

4 Experimental Setup

The datasets used for testing and recognizing emotions are ISEAR dataset (ise), Twitter Emotion Corpus (Mohammad, 2012) and Semeval-2018 Affect in Tweets English Test dataset (Saif M. Mohammad, 2018).

ISEAR dataset (ise) The "International Survey on Emotion Antecedents and Reactions" dataset published by Scherer and Wallbott is built by collecting questionnaires answered by people with different cultural backgrounds (Bostan and Klinger, 2018). A total of 7,665 sentences labeled with single emotions. The labels are joy, fear, anger, sadness, disgust, shame, and guilt.

Twitter Emotion Corpus (Mohammad, 2012) is prepared with emotion-word hashtags as emotion labels. These are termed as noisy labels as labelled by users. This corpus contains 21050 sentences labelled with one of the emotions from Ekman's emotion model.

Semeval-2018 Affect in Tweets English Test dataset (Saif M. Mohammad, 2018) is gold standard multi-labelled dataset with 3259 tweets annotated with multiple emotions. The emotion labels are anger, anticipation, disgust, fear, joy, love, optimism, pessimism, sadness, surprise, trust. Every emotion is labelled as 0 or 1 to show presence of that emotion. If all are 0 then tweet is considered to be neutral.

4.1 Modification in Datasets for Testing

Few modifications are incorporated before using them for testing and experiments.

4.1.1 Mapping of Emotion Labels to Ekman's Emotion Model

Not all of these datasets are labelled with the Ekman's emotions. The researchers follow different emotion models such as the Plutchick model, the Parrot's emotion model. Also few researchers use emotion categorization as per requirement of the system and the data. Hence, we have mapped these emotion labels to one of the best suitable Ekman's emotion as shown in Table-2

This mapping is coarse-grain mapping as the Ekman's model represents six basic emotions - Happy, Surprise, Anger, Disgust, Fear and Sadness and all other emotions can be mapped in these emotions directly.

4.1.2 Labeling Datasets with Sentiment Values

Not all the above mentioned datasets are annotated with sentiment values. Hence, to illustrate this problem definition, we labelled these datasets with sentiment value, based on already available emotion labels to the sentences.

Sr No	Dataset Name	Happy	Surprise	Anger	Fear	Disgust	Sadness
1	ISEAR dataset	Happy	–	Anger	Fear	Disgust	Sadness
2	Semeval-2018 Task-1 Affects in Tweets English Dataset	Anticipation, Joy, Love, Optimism, Trust	Surprise	Anger	Worry	–	Pessimism, Sadness
3	Twitter Emotion Corpus	Joy	Surprise	Anger	Fear	Disgust	Sadness

Table 2: Mapping of original emotion labels to Ekman's emotions

The sentences from datasets with positive emotions such as happy, love, joy, surprise etc are labelled with positive (+1) sentiment value. And the sentences with negative emotions such as anger, disgust, fear, sadness etc are mapped to negative (-1) sentiment value.

Now, the datasets are in the required format for further processing and testing. The format of every testing example is:

$< sentence, sentiment\ value, emotion >$.

While testing the system, the sentence and the sentiment value from modified datasets are considered as input and best possible emotion is recognized. Later, these predicted emotions are compared with these emotion labels for checking the accuracy of the system.

4.2 Experiments

- The sentence is pre-processed to remove the stopwords, hyperlinks, hashtags, usernames and the special characters if any. Also part of speech tagging is done to obtain open-class words that is noun, verb, adjective, and adverb. The NLTK PoS tagger and the Wordnet word categories are used to perform the same. As the closed-class words do not contribute towards emotions, they are not considered for further processing. The syntactic dependencies are retrieved for given input sentence using Stanford coreNLP dependency parser.

- We have experimented on different datsets mentioned in Table-2 using different pre-trained word embeddings such as Google Word2Vec (Tomas Mikolov, 2013), Glove (Jeffrey Pennington and Manning, 2014), and FastText (Joulin et al., 2016)

- The experiments are performed on the text with sentiment information and without sentiment information too. The weighted F-score, precision and recall are used as metrics to evaluate the accuracy of system.

5 Results and Discussions

As shown in Table-3, the experiments are performed in two different ways, first by considering only text/sentences as input and secondly by considering text and its sentiment value as input. The experiments conducted with only sentences as input, serves here as baseline against experiments using sentences with sentiment information.

The sentence with respective sentiment information as input shows significant improvement in weighted F-score value. The results are shown in the Table-3. It is observed that Google Word2Vec word vectors performs better than other word embedding.

The Semeval-2018 Task-1 dataset (Saif M. Mohammad, 2018) is multi-labelled dataset. The annotated emotions are assigned independently. This task is multi-class emotion recognition so we consider prediction 'correct' even if one of the assigned emotions is predicted by our system.

It is visible in Table-4 that sentences with sentiment value as input, the F-score of every individual emotion category has been improved drastically. This shows better prospects for such emotion recognition and conversion process for new resource creation with fine-grained labeling from the sentiment to the emotion.

The recall values for datasets using Google Word2Vec are shown in Table-5 for illustration.

Sr No	Word Embedding	ISEAR dataset		Twitter Emotion Corpus		Semeval-18 Task-1 Dataset	
		w/o Sentiment Value	with Sentiment Value	w/o Sentiment Value	with Sentiment Value	w/o Sentiment Value	with Sentiment Value
1	Google Word Vectors	0.37	**0.56**	0.32	**0.52**	0.52	**0.76**
2	GLOVE vectors	0.23	0.33	0.25	0.45	0.38	0.67
3	Fast Text Word Vectors	0.34	0.49	0.30	0.49	0.51	0.74

Table 3: Weighted F-score using different word embedding and with / without Sentiment Value

Sr No	Method (Input)	Anger	Disgust	Fear	Happy	Sadness	Surprise
	ISEAR Dataset						
1	Sentence	0.35	0.27	0.43	0.45	0.33	–
2	Sentence and Sentiment Value	**0.45**	**0.39**	**0.51**	**0.97**	**0.52**	–
	Twitter Emotion Corpus						
3	Sentence	0.21	0.11	0.28	0.55	0.14	0.10
4	Sentence and Sentiment Value	**0.32**	**0.18**	**0.42**	**0.79**	**0.55**	**0.15**
	Semeval-2018 Task-1 Affects in Tweets Dataset						
5	Sentence	0.40	0.43	0.39	0.66	0.51	0.21
6	Sentence and Sentiment Value	**0.67**	**0.65**	**0.53**	**0.92**	**0.76**	**0.32**

Table 4: Emotion category-wise F-score for emotion recognition using Google Word2Vec vectors

Recall values in case of the method with sentiment information has increased by approximately 50% than method without sentiment values. This shows significant improvement in correctly predicting emotion on use of sentiment information.

The Table-6 illustrates the confusion matrices for results of emotion recognition on ISEAR dataset. It can be seen that positive emotions and negative emotions are rarely confused with each other by using sentiment information. The recall and precision is also increased for every emotion. Yet emotions belonging to the same sentiment value need to be achieved with better accuracy. The emotion 'surprise' is not part of emotion labels for ISEAR so 0s in row for 'surprise'.

Conclusion

The proposed system suggests the way for creation of a resource from the available resources. The use of more easily available sentiment labelled data for creating emotion annotated data is significant. The use of sentiment information for recognizing the emotion is good example of fine-grain labeling task.

The proposed approach shows much better accuracy for text labelled with sentiment value than the baseline as text without sentiment information. The use of sentiment information helps to segregate at initial level between emotions with the different polarity.

As the word vectors are based on distributional hypothesis, they may have higher cosine similarity for opposite words, for example, 'happy' and 'sad'. The synonyms may have very low cosine similarity value. This can affect overall accuracy of the system. The rare words may not contribute much and very common words may get very high cosine similarity with opposite words too. Hence, it is necessary to select better list of emotion-specific words. More processing and linguistic information may be added to improve the accuracy of this system.

Sr No	Method (Input)	Anger	Disgust	Fear	Happy	Sadness	Surprise
	ISEAR Dataset						
1	Sentence	0.29	0.19	0.37	0.81	0.26	–
2	Sentence and Sentiment Value	**0.43**	**0.26**	**0.52**	**0.94**	**0.65**	–
	Twitter Emotion Corpus						
3	Sentence	0.22	0.12	0.21	0.78	0.10	0.10
4	Sentence and Sentiment Value	**0.37**	**0.20**	**0.34**	**0.94**	**0.55**	**0.10**
	Semeval-2018 Task-1 Affects in Tweets Dataset						
5	Sentence	0.30	0.33	0.49	0.76	0.45	0.89
6	Sentence and Sentiment Value	**0.59**	**0.57**	**0.70**	**0.92**	**0.82**	**0.92**

Table 5: Emotion category-wise Recall values for emotion recognition using Google Word2Vec vectors

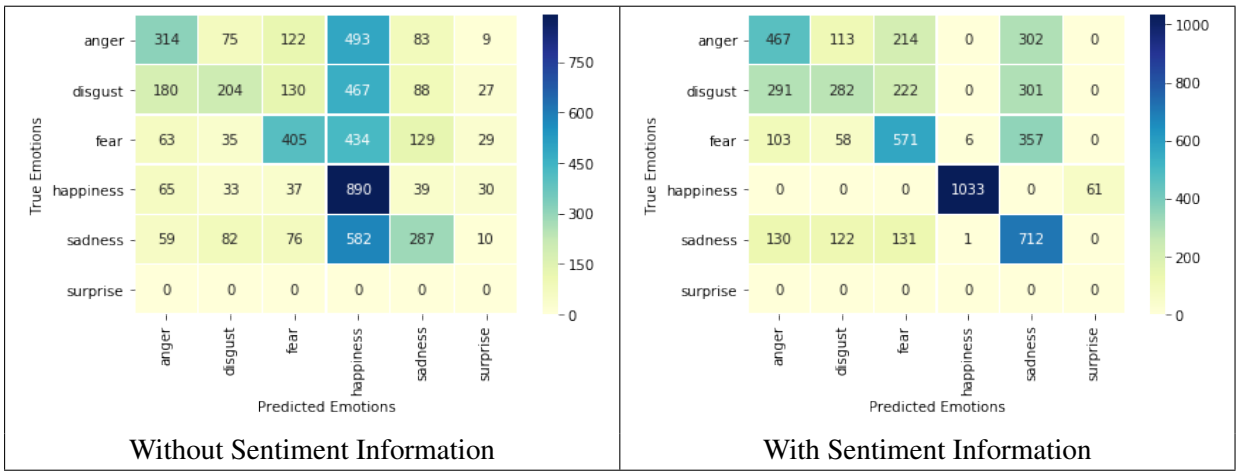


Table 6: Confusion Matrix for ISEAR dataset using Google Word2Vec Vector

References

Aijun An A Agrawal. 2012. Unsupervised emotion detection from text using semantic and syntactic relations. *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology*, 1:346–353.

Manos Papagelis Ameeta Agrawal, Aijun An. 2018. Learning emotion-enriched word representations. *The 27th International Conference on Computational Linguistics, Santa Fe, New Mexico, USA*, page 950–961.

Wlodek Zadrozny Armin Seyeditabari, Narges Tabari. 2018. Emotion detection in text: a review. *arXiv:1806.00674v1 [cs.CL], 2nd Jun 2018*.

Laura-Ana-Maria Bostan and Roman Klinger. 2018. An analysis of annotated corpora for emotion classification in text. *The 27th International Conference on Computational Linguistics,Santa Fe, New Mexico, USA*, page 2104–2119.

Yi-Shin Chen Carlos Argueta, Elvis Saravia. 2015. Unsupervised graph-based patterns extraction for emotion classification. *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 336–341.

Al Moatassime Hassan Douiji yasmina, Mousannif Hajar. 2016. Using youtube comments for text-based emotion recognition. *The 7th International Conference on Ambient Systems, Networks and Technologies (ANT)*, pages 292–299.

Nancy Ide Jared Suttles. Distant supervision for emotion classification with discrete binary values. *CICLING, March 2013*.

Richard Socher Jeffrey Pennington and Christopher D. Manning. 2014. Glove: Global vectors for word representation.

Jing Jiang Pradeep Karuturi-William Brendel Jianfei Yu, Luıs Marujo. 2018. Improving multilabel emotion classification via sentiment classification with dual attention transfer network. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1097–1102.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Weiyuan Li and Hua Xu. 2014. Text-based emotion classification using emotion cause extraction. *Expert Systems with Applications*, 41(4):1742–1749.

Jonathan Juncal-Martínez Enrique Costa-Montenegro Milagros Fernández-Gavilanes, Tamara Àlvarez-López. 2015. Gti: An unsupervised approach for sentiment analysis in twitter. *9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 533–538.

Kiritchenko Svetlana Mohammad, Saif M. 2018. Understanding emotions: A dataset of tweets to study interactions between affect categories. In *11th Edition of the Language Resources and Evaluation Conference (LREC),Miyazaki, Japan*.

Saif Mohammad. 2012. emotional tweets. *The First Joint Conference on Lexical and Computational Semantics (*Sem), Montreal, Canada.*, page 246–255.

Mohammad Salameh-Svetlana Kiritchenko Saif M. Mohammad, Felipe Bravo-Marquez. 2018. Semeval-2018 Task 1: Affect in tweets. *The International Workshop on Semantic Evaluation (SemEval-2018),New Orleans, LA, USA*, page 246–255.

Rong Wang Shoushan Li, Lei Huang and Guodong Zhou. 2015. Sentence-level emotion classification with label and context dependence. *53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 1:1045–1053.

Kai Chen Greg Corrado-Jeffrey Dean Tomas Mikolov, Ilya Sutskever. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*.

A Metric for Lexical Complexity in Malayalam

Richard Shallam (imrichsham@gmail.com)
Independent Researcher

Ashwini Vaidya (avaidya@hss.iitd.ac.in)
IIT Delhi

Abstract

This paper proposes a metric to quantify lexical complexity in Malayalam. The metric utilizes word frequency, orthography and morphology as the three factors affecting visual word recognition in Malayalam. Malayalam differs from other Indian languages due to its agglutinative morphology and orthography, which are incorporated into our model. The predictions made by our model are then evaluated against reaction times in a lexical decision task. We find that reaction times are predicted by frequency, morphological complexity and script complexity. We also explore the interactions between morphological complexity with frequency and script in our results. To the best of our knowledge, this is the first study on lexical complexity in Malayalam.

Keywords: lexical processing, visual word recognition, lexical complexity, Dravidian languages

1 Introduction

The task of visual word recognition is related to language processing at the level of a word/lexical item. A word can be analyzed at several linguistic levels, and the word recognition task helps us understand the role of these levels in relation to processing, memory and attention. In psycholinguistics, previous work on this topic focuses on understanding the individual variables that affect the lexical processing of words. If we can quantify the influence of variables ranging from orthographic features to semantic factors on the cognitive processing of words, it would help us in understanding the critical factors underlying visual word recognition (and pattern recognition, more generally). The resulting model of word recognition can be evaluated against human judgements.

Models of word recognition are especially relevant for eye-tracking studies, where they have been extensively explored (Rayner and Duffy, 1986). Word recognition models have also been used to understand reading disabilities such as phonological and surface dyslexia (Balota et al., 2006). For these studies, it is crucial to tease apart the effect of various factors that affect the task of reading. Previous research has shown that the eye gaze duration is affected by frequency, orthography, morphology and phonology, among others. Apart from these studies, an understanding of lexical complexity is also an interesting topic for study on its own.

In this paper, we explore the case of Malayalam and in particular examine three factors that could predict word complexity in the language: frequency, orthography and morphology. The role of variables that determine word recognition in Malayalam has not been explored, as it has been for Hindi (Husain et al., 2015; Verma et al., 2018). Quantifying these factors in a model of lexical complexity can help us in developing norms that are useful in areas such as reading studies and word generation for lexical decision tasks. Further, this would contribute towards cross-linguistic comparison of these factors from a different language family. To the best of our knowledge, this is the first work that examines lexical complexity in Malayalam.

2 Lexical Complexity

The task of visual word recognition involves the cognitive processing of visual information and comparing it with a particular internal

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 178–183
Hyderabad, India, December 2019.

mental representation of a word. This representation itself may be at the graphemic, phonemic, morphemic and lexical semantic level, all of which have been shown to affect word recognition (Balota et al., 2006). In the sections that follow, we describe the three factors that are included in our study.

2.1 Word Frequency

The effect of word frequency is robust and has been well studied across word recognition tasks (Balota et al., 2006). High frequency words tend to be recognized faster than low frequency words. In eye tracking studies high frequency words have lower gaze duration and fixation measures. We would expect that frequency would have a similar effect on the Malayalam data, where high frequency would contribute towards a lower lexical complexity.

2.2 Morphology

A word may be composed of a single morpheme e.g. *boy* or more than one e.g. *funnily: funny+ ly*. The role of morphology in word recognition is at a sub-lexical level. Morphology as a measure is particularly relevant for an agglutinative language such as Malayalam, which also exhibits productive word compounding e.g. Just the word മരം (mara) "tree" has a number of morphological forms such as

മരത്തിൽ (marattil) - in the tree
മരത്തിന്റെ (marattin̲re) - of the tree
മരങ്ങൾക്കിടയിലൂടെ (maraṅṅaḷkkiṭayilūṭe) - through the trees
മരക്കൊമ്പുകൾ (marakkeāmpukaḷ) - tree branches

Early studies that looked at the effect of morphology on lexical access have suggested that polymorphemic words (i.e. words consisting of more than one morpheme) are decomposed into their component parts during online processing. This process would find the root first (e.g. *funny* and on finding it, proceed to search stored affix-stem combinations till *funnily* is retrieved (Taft and Forster, 1975). In a morphologically-rich language such as Malayalam, we would expect that this would be an important factor in lexical processing.

2.3 Orthography

The visual processing of words involves processing at the orthographic level as well. This implies that the writing system of various languages will influence recognition. A writing system–whether alpha-syllabic, logographic or alphabetic has been shown to influence reading times (Katz and Frost, 1992). Sub-lexical properties such as letter features and their interactions with the words themselves can also influence word complexity, which needs to be accounted for in the model.

3 Method

In order to compute the lexical complexity metric, token frequency, morphology and orthography were included as our variables. Below, the methods for computing the values for each of these variables are discussed.

3.1 Corpus

In order to compute our metric for Malayalam, we first obtained a corpus from the Leipzig Corpora Collection containing 300,000 sentences from Malayalam Wikipedia articles and 100,000 sentences from Malayalam news crawl (Goldhahn et al., 2012). The corpus was then preprocessed by removing punctuation and special characters, and then tokenized using whitespace. The text was also normalized to remove inconsistencies in spelling using the Indic NLP Library[1] and this resulted in 4,711,219 tokens and 762,858 unique types.

3.2 Word Frequency Metric

The corpus was used to collect counts for each word and then scaled them between 0 and 1, which was then inverted such that the most frequent tokens have a value closer to 0 and the less frequent tokens will have a value approaching 1. This score indicated the relative frequency of each word in this corpus, and the idea that highly frequent words are much easier to process than those that have lower frequency.

3.3 Morphology Metric

Our morphology metric required us to obtain information about the root and the morpho-

[1] https://anoopkunchukuttan.github.io/indic_nlp_library/

logical affixes for a given word. Given the rich morphology and compounding processes in the language, we had to make use of a two-step process to compute our scores.

First, *SandhiSplitter* (Devadath et al., 2014) was used to split tokens that are compound words into their constituent component words. For example, consider the compound word കാരണമായിരിക്കണം (kAraNamAyirikkaNaM)

കാരണമായിരിക്കണം ⇒ കാരണം + ആയിരിക്കണം

kāraṇamāyirikkaṇaṁ ⇒ kāraṇaṁ + āyirikkaṇaṁ

"must be the reason" ⇒ "reason" + "must be"

As a second step, these results were passed through IndicStemmer[2], a rule-based stemmer for Malayalam, which further decomposed the words into stems and affixes. As an example, the word ലേഖനങ്ങളുടെ (lēkhanaṅṅaḷuṭe) meaning "Of articles". is decomposed into the stem ലേഖനം (lēkhanaṁ) meaning article with the suffix -ങ്ങൾ (ṅṅal) indicating plural and --ുടെ (uṭe) indicating the Genitive case. In our metric we only considered suffixes as in Malayalam usually contains always suffixes being added to the end of the stem.

After this two-step process, we are able to obtain the stems and suffixes for a given word.

Morpheme Count

By simply summing the number of stems and suffixes, the total number of morphemes contained in each word is computed. For example, the word സമ്പത്സമൃദ്ധിയും (sampat-samrd'dhiyuṁ) meaning "prosperity" is a compound word split into constituent words സമ്പത്ത് (sampatt) meaning "richness" and സമൃദ്ധിയും (samrd'dhiyuṁ) meaning "and plentiful". സമൃദ്ധിയും (samrd'dhiyuṁ) is further stemmed to stem word സമൃദ്ധി (samrd'dhi) meaning "plentiful" and suffix -ും (uṁ) meaning "-and". സമ്പത്ത് (sampatt) is a root word. Thus, the number of morphemes in this case is three, counting the two stems and one suffix.

Based on this pre-processing, we then calculate the total number of morphemes for each whole word and then scale this number between 0 and 1 to give a morpheme score. We note that there could be several different ways to compute the morpheme score, as affixes themselves are not all alike. In this preliminary study, it was not immediately apparent how the differing costs for various affixes could be calculated. Additionally, fine-grained information regarding the morphological properties of the affixes (e.g. whether they were inflectional or derivational) was not easily obtained with existing tools and resources. In future work, we plan to explore this possibility by enhancing the morphological analyzer's output.

[2]https://github.com/libindic/indicstemmer

3.4 Orthography Metric

Malayalam is an alphasyllabic writing system that has its source in the Vatteluttu alphabet from the 9th century. Its modern alphabets have been borrowed from the Grantha alphabet. It consists of 15 vowels and 36 consonant letters.

We devised a script score based on complexity of the script in the following three ways:-

Mismatch in Spoken and Visual Order

In the alpha-syllabic script of Malayalam, vowels may either appear as letters at the beginning of a word or as diacritics. Consonants themselves are understood to have an inherent schwa, which is not separately represented. The diacritics will appear either left or right of the consonant it modifies. If it appears to the left, there will be a discrepancy in the phonemic and the orthographic order, as the vowel will always be pronounced after the consonant, but read before the consonant actually appear in the text. For example:

ക + െ = കെ

ka + .e = ke

Here the vowel violates the order in which it is spoken. Similarly: ക + േ = കേ (ka + ē = kē), as seen in കേൾക്കുക (kēḷkkuka) meaning "hear". Such inconsistencies in spoken and visual order have been shown to incur a cost in Hindi word recognition (which is also an alpha-syllabic script) (Vaid and Gupta, 2002).

In order to capture the lexical processing cost for such a discrepancy, we give a penalty of 1 every time it occurs in the word.

Diacritic Appearing Above or Below

In Malayalam, the diacritic may also appear above or below a consonant. In such a case, we we give a penalty of 0.5 to the word. For example the symbol ് also known as *virama* is used to replace the inherent schwa sound of consonants with ŭ. As in ക + ് = ക് (ka + virama = ku)

Ligatures and Consonant Clusters

A penalty of one is assigned for every two letters that form a composite glyph. For example: മന്ത്രി (mantri) = മന് + ത്രി (man + tri) where the new composite glyph is ന്ത (ntra).

With the above complexity rules in place, the total penalty cost for each whole word is calculated. Then the total penalty for each word is scaled linearly to between 0 and 1 to give us an orthographic score.

3.5 Evaluation of the Complexity Metric

In order to evaluate our lexical complexity metric, we used a lexical decision task paradigm to collect reaction times for a sample of Malayalam words. More complex words would result in longer reaction times, and vice versa. This would help us evaluate whether our lexical complexity model could predict reaction times for the given set of words.

We used a well-understood experimental paradigm in the form of a lexical decision task. In such a setup, a participant will see a word stimuli on a screen which they have to classify as either a word or a non-word using a button press. The response time (RT) is calculated from the point the word appears on the screen to the point where the participant presses the response button.

Materials

Our task consisted of a balanced set of 50 Malayalam words and 50 pseudowords. Pseudowords follow the phonotactics of the language, but have no lexical meaning (i.e. are not legitimate words). In order to select words for the task, two sets of 25 words were randomly sampled from the unique tokens obtained from the Leipzig Corpus. The first set was randomly sampled from words with a frequency score between the range of 0.1 to 0.4 to obtain high frequency words as calculated by the metric. The second set was chosen similarly but with frequency score between the range of 0.7 to 0.9 to yield low frequency words. If the sampled word turned out to be an English word written in Malayalam or happens to be a proper noun, it was replaced with another until both sets had 25 words each.

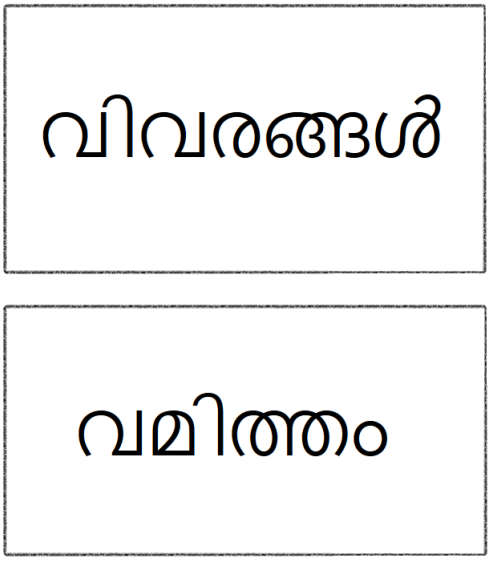


Figure 1: Stimuli word shown for 2500ms. The first word is a proper Malayalam word ("vivaraṅṅaḷ" meaning "information") hence the correct response is to press the 'a' key. The second word is non-word (vamittaṁ) and therefore, the correct response is to press 'l' key.

The pseudowords were constructed in keeping with the phonotactics of Malayalam. Both the pseudowords and the valid words were constrained in length between 6 and 14 characters. Note that we do not take into consideration the reaction times for the pseudowords; they are simply distractors for the participants.

Participants

Participants included 38 students from S.N. College, Kerala, who volunteered for the study. Participants included 20 females and 18 males between the ages of 18 and 23 (mean age of 19.7). All participants were native speakers of Malayalam and had formal education in Malayalam upto grade 10.

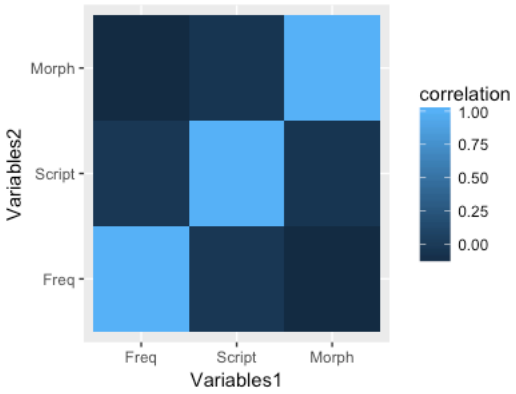


Figure 2: Heat plot showing correlation between the three variables in our test data

	Estimate	*Std. Error*	*t-value*	*p-value*
(Intercept)	4.30	0.679	6.35	0
Script	9.157	3.76	2.43	**0.015 ***
Freq	2.87	0.96	2.97	**0.003 ****
Morph	1.91	0.71	2.67	**0.007 ****
Script:Freq	-3.171	5.77	-0.55	0.58
Script: Morph	-7.64	4.1	-1.873	**0.06 .**
Freq: Morph	-1.79	1.03	-1.743	**0.08 .**
Script:Freq:Morph	0.28	6.31	0.045	0.96

Table 1: Results for all three variables and their interactions. Script and Morphological Complexity as well as Frequency and Morphological Complexity show a significant interaction

Procedure

Participants were tested individually on a computer running the lexical decision task on the JsPsych stimulus presentation software (De Leeuw, 2015). Each participant was asked to press either the 'a' key or the 'l' key for word and non-word respectively. The order of words and pseudowords was randomized for each participant. Participants were instructed to read the word presented and respond with the appropriate button press. Each trial consisted of a word that was presented for 2500ms. A fixation cross was placed in the center for 1600ms between each trial. The first 10 trials were practice trials from a word set different from the study. This enabled participants to get familiarized with the task.

4 Results

The trials belonging to those who scored below 70% in word-non-word accuracy were excluded, which brought the number of participants to 35.

We fit a linear model using the `lm` function in R. Log reaction times were used with frequency, script and morph as the covariates. Figure 2 shows that the three variables are not highly correlated in our test set.

Table 1 shows the results of the regression analysis. The main inference we can draw from the result is that the variables Script, Morphology and Frequency have a significant effect (all p-values < 0.05) on (reaction times) RTs, such that a high cost of script, morph and frequency leads to higher RTs.

In addition, the results also indicate a marginal interaction between Script and Morphology (p=0.06), such that an increase in the script complexity leads to larger increases in RTs for morphologically simpler words (Cost <0.9) compared to morphologically complex words (Cost >0.9) (see Figure 3). There is also a marginal interaction between Morphology and Frequency (p=0.08) such that an increase in the frequency cost leads to higher reaction times in morphologically complex words as compared to morphologically simpler words (see Figure 4).

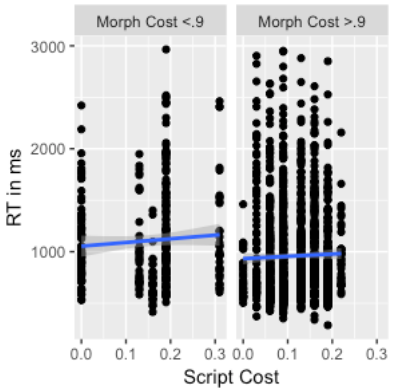


Figure 3: Interaction between Morphological Complexity and Script Complexity

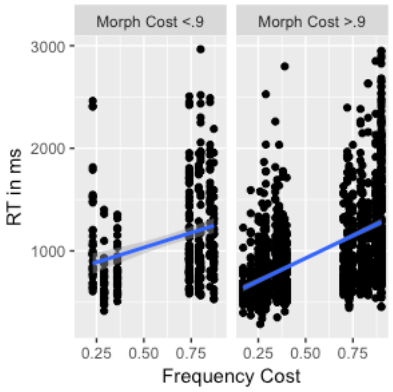


Figure 4: Interaction between Morphological Complexity and Frequency Cost. Note that a low Frequency Cost corresponds to a high Frequency Count for a word

5 Discussion

Our results replicate the robust effects of frequency on lexical processing in Malayalam. As frequency is a known predictor of reaction times, we expected to find a significant effect for frequency, but we particularly wanted to understand the effect of morphology and orthography on word recognition in Malayalam. Orthographic complexity as captured by diacritic placement and ligatures also has a significant effect on lexical processing. Similarly, we also find an effect for morphological complexity in terms of the number of morphemes in a word.

The interactions in our model point to an interesting relationship between high frequency words and morphological complexity. It appears that the effect of frequency cost becomes more pronounced in more complex words. In other words, low frequency words lead to higher reaction times particularly when they are morphologically complex. Perhaps this is because the cost of lexical decomposition is higher in these words. On the other hand, the effect size of script is weaker and becomes visible only when the word is morphologically simple. When the word is morphologically complex, this effect is not very apparent.

This work points to many interesting future avenues for exploring lexical complexity in an agglutinative language like Malayalam. Particularly, the effect of morphological complexity on factors like frequency need to be explored more thoroughly. In the future, we plan to carry out experiments with a larger set of items for the lexical decision task, as this was a preliminary study. We also plan to experiment with other measures of morphological complexity that take into account information about the type as well as the number of morphemes.

References

David A Balota, Melvin J Yap, and Michael J Cortese. 2006. Visual word recognition: The journey from features to meaning (a travel update). In *Handbook of Psycholinguistics*, pages 285–375. Elsevier.

Joshua R De Leeuw. 2015. jspsych: A javascript library for creating behavioral experiments in a web browser. *Behavior research methods*, 47(1):1–12.

VV Devadath, Litton J Kurisinkel, Dipti Misra Sharma, and Vasudeva Varma. 2014. A sandhi splitter for malayalam. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 156–161.

Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *LREC*, volume 29, pages 31–43.

Samar Husain, Shravan Vasishth, and Narayanan Srinivasan. 2015. Integration and prediction difficulty in hindi sentence comprehension: Evidence from an eye-tracking corpus. *Journal of Eye Movement Research*, 8(2):1–12.

Leonard Katz and Ram Frost. 1992. The reading process is different for different orthographies: The orthographic depth hypothesis. In *Advances in psychology*, volume 94, pages 67–84. Elsevier.

Keith Rayner and Susan A Duffy. 1986. Lexical complexity and fixation times in reading: Effects of word frequency, verb complexity, and lexical ambiguity. *Memory & cognition*, 14(3):191–201.

Marcus Taft and Kenneth I Forster. 1975. Lexical storage and retrieval of prefixed words. *Journal of verbal learning and verbal behavior*, 14(6):638–647.

Jyotsna Vaid and Ashum Gupta. 2002. Exploring word recognition in a semi-alphabetic script: The case of devanagari. *Brain and Language*, 81(1-3):679–690.

Ark Verma, V. Sikarwar, H. Yadav, J. Ranjith, and Pawan Kumar. 2018. Shabd: A psycholinguistics database for hindi words. In *Proceedings of ACCS 2018*.

Kunji : A Resource Management System for Higher Productivity in Computer Aided Translation Tools

Priyank Gupta, Rashid Ahmad, Manish Shrivastava, Dipti Misra Sharma
LTRC, KCIS
IIIT-Hyderabad
{priyank.gupta, rashid.ahmadpg08}@research.iiit.ac.in
{m.shrivastava,dipti}@iiit.ac.in

Abstract

Complex NLP applications, such as machine translation systems, utilize various kinds of resources namely lexical, multiword, domain dictionaries, maps and rules etc. Similarly, translators working on Computer Aided Translation workbenches, also require help from various kinds of resources - glossaries, terminologies, concordances and translation memory in the workbenches in order to increase their productivity. Additionally, translators have to look away from the workbenches for linguistic resources like Named Entities, Multiwords, lexical and lexeme dictionaries in order to get help, as the available resources like concordances, terminologies and glossaries are often not enough. In this paper we present Kunji, a resource management system for translation workbenches and MT modules. This system can be easily integrated in translation workbenches and can also be used as a management tool for resources for MT systems. The described resource management system has been integrated in a translation workbench Transzaar. We also study the impact of providing this resource management system along with linguistic resources on the productivity of translators for English-Hindi language pair. When the linguistic resources like lexeme, NER and MWE dictionaries were made available to translators in addition to their regular translation memories, concordances and terminologies, their productivity increased by 15.61%.

1 Introduction

NLP applications - machine translation systems and translation workbenches (which can have multiple MT systems integrated), are complex in nature as they are built with various complex heterogeneous NLP modules. These complex NLP applications are compute and knowledge intensive in nature and require various types of resources at different levels of processing.

The translators using the translation workbenches have to look for various resources - glossaries, terminologies, concordances and translation memory in order to supplement their translation tasks. It is still hard to maintain consistency while maintaining high productivity. Sometimes, the aid the translators get from the resources is not enough and they have to look for linguistic resources - Named Entities, lexical, bilingual, Multiwords and lexeme dictionaries - offline or online for the correct and appropriate meaning of a given phrase or word leading to additional cognitive load on translators.

In MT system, when MT modules execute on the given input, various types of resources namely lexical, bilingual, domain, multiword dictionaries, paradigms, maps files and rules are required at various stages of the processing. With the exploration of Sampark and Anusaaraka MT systems (ILMT, 2009; Chaudhury et al., 2010), it has been found that the resources can be used in a more efficient way in the both systems. These resources are developed by by NLP researchers who face resource management issues like exporting, importing, storage *etc*. For a large scale NLP application development, inefficient storage and management of these resources can become a bottleneck for productivity.

For the translation tasks and MT applications' development for Indian languages, the above mentioned issues become more critical as the digital content available in Indian languages are way less compared to other languages. As Hindi is one of the top 5 spoken languages of the world but still its digital content available on the web is less than 0.05% while the content of the other Indian languages are still lower. Hence the need of a resource management system arises which can ad-

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 184–192
Hyderabad, India, December 2019.

dress the issues of both translators as well as MT modules' development.

In this paper, we present a resource management system named Kunji, which addresses the issues related to resources of both translation workbenches and MT system development for Indian languages. It can be used in translation workbenches to facilitate the translators to use additional linguistic resources - domain, lexical, named entities, multiword and WSD dictionaries - along with the terminologies, concordances and glossaries. It facilitates the translators in two ways - i) reduction of their repetitive load by providing searching and filter mechanism and ii) reuse of their previously translated words, phrases and sentences by providing the provision of personal and domain dictionaries. It shifts the problem of recall to recognition which causes less cognitive load on translators by providing them provision to search the corresponding terms and recognize. It also facilitates the NLP researchers to manage the resources for MT modules in a robust manner while addressing their issues of management, export, import, formats and efficient search etc. It also lets them evaluate and verify their work from other senior expert. Such system facilitates the large scale MT applications development by making the resources available to multiple modules and their facilitation to process and reuse them efficiently in an MT system.

Additionally we performed an experiment to study the impact of providing this resource management system along with linguistic resources on the productivity of translators. The described resource management system, Kunji, has been integrated in the translation workbench Transzaar (Ahmad et al., 2018). The resources with linguistic features like lexeme, NER and MWE dictionaries have been provided in Kunji to facilitate translators in addition to their regular translation memories, concordances and terminologies for English-Hindi language pair. We observed that the productivity of the translators increased by 15.61% when the resources with linguistic features were provided to the translators in addition to their regular resources (translation memories, glossaries, concordances and terminologies).

In this paper, background and motivation for Kunji is described in Section-2 followed by related work in Section-3. The detailed architecture and functionality of Kunji is described in Section-4. Section-5 describes the experiment performed on Kunji followed by its results and discussion in Section-6. In Section-7, we conclude our work.

2 Background and Motivation

In current scenario, various translation workbenches(CAT tools) are available. Some of them offer the resources like translation memories(TM), concordances, terminologies and glossaries to translators but there is a lack of the resource management systems which provide the provision by which a translator can use additional linguistic resources like NE, MWE and lexeme dictionaries. Additionally there is a lack of the provisions of reuse, management and verification of these resources which can lead a process to improvement and development of the MT systems.

There are several tools for resource development but they are built with separate purposes and languages as they are for task specific. Like for annotators they have separate tools in Indian languages like Sanchay (Singh, 2008) but it is desktop based application. Some tools like GATE (Cunningham, 2002) addresses the some of the problems faced by NLP researchers while developing NLP resources but it is desktop based application.

It motivated us to develop Kunji, a resource management tool, which facilitates the translators in translator workbenches as well as MT system development process. It tries to address the issues faced by both translators as well as MT module developers. It allows the translators to use and manage the linguistic resources along with their terminologies and glossaries.

3 Related Work

There are some translation workbenches with separate resource management systems such as: Smartcat [1], Memsource [2] and Matecat (Federico et al., 2014) are web based CAT tools, provide central resources management for a project but they do not provide the provision for the linguistic resources. CASMACAT (Alabau et al., 2014) is a translation workbench which is web based and offers advanced functionality for computer-aided translation. It offers TMs but it also lacks the provision for the linguistic resources. PET (Aziz et al., 2012) is a tool to postedit to for evaluating

[1] https://www.smartcat.ai
[2] https://www.memsource.com

the quality of translations. It evaluates the efforts required in translations in order to be fixed. SDL Trados, MemoQ[3] and Anubis (Jaworski, 2013) are CAT tools which are not web based and both lack the provision to use linguistic resources.

Brat (Stenetorp et al., 2012) is basically a web based annotation tool which focuses mainly on text annotations to enhance annotators productivity by closely integrating NLP technology into the annotation process. It doesnt address the issues in resources management with respect to translators and large scale NLP application development. Creating language resources for NLP in Indian languages (Sangal and Sharma, 2001) defines a novel idea in which the development of lexical resources is linked with an example NLP application like MT then it can act as a test bed for the developing resources and provide constant feedback. Sanchay (Singh, 2008) provides an extra layer of easily customizable language encoding support for less computerized languages along with an editor named Sanchay with different types of fonts and language encoding supports but it is desktop based application. GATE (Cunningham, 2002) enables users to develop and deploy the language engineering tools and resources in a robust manner. It is a desktop based tool which supports many NLP and information extraction tasks in multiple languages. Nancy et al (Ide and Romary, 2009) presents an abstract data model and its implementation for linguistic annotations. Annomarket (Dimitrov et al., 2014) described a cloud-based open platform for text mining, which aims to assist the development and deployment of robust, large-scale text processing applications.

We see that in some translation workbenches, the resource management systems for TM, Terminologies and glossaries are available but they don't provide the provision for linguistic resources as well.

4 Kunji : Resource Management System

We propose Kunji-a Resources management System, which is a web based system based on microservices architecture. It can be easily integrated to a translation workbench and provides a mechanism to facilitate translators to use and manage the various types of resources (i.e. like terminologies, glossaries and domain dictionaries along with linguistic resources(lexeme dictionaries, NERs, MWEs etc.)). It does not only facilitate the translators but also facilitates the MT modules development by facilitating language researchers to manage the language engineering resources in a robust manner for various language processing tasks and evaluate them which boosts the process of large scale MT development.

4.1 Microservices Based Architecture and NLP Applications

Microservices (Thönes, 2015) based approach is an architectural concept of developing an application or software systems according to which a functionality or process can be developed, deployed and tested independently. An application is divided into set of such modular components or functionalities in which each functionality is an independent or disjoint from the others' in the application. In this approach, each functionality or component has a proper structured interface which is called API and is used to communicate with the corresponding module. It helps to overcome the drawbacks of monolithic approach.

The complex NLP applications- especially MT systems or translation workbenches are knowledge and compute intensive in nature. So management and development of the resources for them requires deep knowledge of corresponding domain, nature of language while implementing algorithm for it requires knowledge of both streams NLP and computer science. For the development of the resource management tools for a translation workbench, the requirements of translators need to be understood. Many existing NLP applications for resource management follow monolithic design and hence static in nature. In NLP, for building scalable, distributed resource management systems , microservices based architecture can overcome the barriers of the monolithic architecture.

4.2 Architectural requirements

Various NLP applications - CAT tools have been analyzed for their resource management, their procedures to utilize different kinds of resources. For the translation workbenches, various aspects like their reuse, verification and sharing of resources in a projects among various translators which are linked to a project. In MT modules, the procedures to access, management and reuse of the resources have been analyzed along with their functional and development requirements. We figured out the fol-

[3] https://www.memoq.com/en

lowing Architectural requirements for the system which are given below:

4.2.1 Microservices based architecture and Web based tool:

The need of microservices based architecture arise to overcome the barriers imposed by monolithic architecture which also facilitates the creation of the dynamic web based tool for the same. In such paradigm, every functionality of the resource management system will be exposed as an independent service which can be utilized by the resource management in translation workbenches in order to facilitate the translators. It can also facilitate the NLP resource management for large scale MT development.

Today is the era of the web, everything we see and work on is on web or is exposed as a web based service or tool. Due to mobile and internet evolution, many people are internet friendly even non-technical people. So in order to make people use and develop resources in more efficient way it is a major need to make it available in web as a form of web-tool which will also makes the application platform independent.

4.2.2 Facilitation in translation workbench:

The resources management system should be designed in such a way so that it should be able to integrate in the translation workbench in order to facilitate translators to increase their productivity. So that translators can manage and use their own private dictionaries, domain dictionaries, terminologies among with the project or task specific dictionaries.

4.2.3 Support for linguistic resources in translation workbenches:

It should support the translators to use, reuse and manage the linguistic resources like NE, MWE and lexeme dictionaries dictionaries as well.

4.3 Our Architecture

We explored the various aspects of NLP resources management and the ways in which they can be utilized in a translation workbenches to aid the translation tasks to translators as well as their facilitation to NLP researchers and annotators towards development of large Scale MT systems. Taking such requirements in consideration, we designed the architecture of the system such that it is a collection of independent microservices for each of the functionality which can be deployed independently. We resolve the problem of monolithic architecture by exposing each functionality as a microservice which can be executed and interacted via RESTful API.

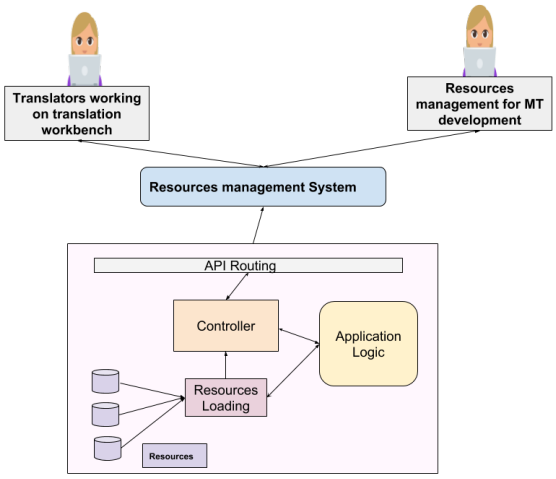


Figure 1: Kunji : Architecture

4.3.1 Kunji : Architecture Explanation

The architecture has been shown in Fig. 1 following with it's explanation.

- **Application logic :** It is the block or portion containing the complete functionality of the system. Hence the system's functionalities are structured, designed and implemented in the functional way independently.
- **In-Memory Resource Loading :** The different types of resources are required in different modules of MT as well as for providing help in the translation workbenches. Their loading from database at each execution process makes process slow as well as it imposes an unnecessary load to the system. Hence the services of the system are designed in such a way that resources load at the time of the starting of the service not at the time of invocation of the service.
- **REST APIs :** REST(Representational state transfer) API is basically an approach or technology for communications which is used in developing web services. It provides many methods such as HTTP or HTTPs for implementation. We expose each of the functionality of the resource management system as a

microservice with REST API interface to interact with. The APIs are created for create, add, import, export, delete, link, update and verify.

- **UI** The simple UI has been created with the HTML5 and JavaScript

4.3.2 Structure of Resources

First the configuration of a resource is defined followed by the corresponding entries of the data inside the resource as follows:

- Configuration of a resource : It contains the following fields as given below in the Table-1:

Resource Configuration Fields
resourceId
resourceName
resourceType
domain
subDomain
srcLang
tgtLang
project
client
description
createdBy
modifiedBy
creationDatetime
lastModifiedDateTime

Table 1: Resource configuration.

First the configuration of the resource is created as the configuration mentioned in Table-1:

- A data record of a resource will look like as mentioned in Table-2 :

  It contains the record with its target value and category and other required fields.

- **Explanation:** First the configuration of a particular resource type is created then its corresponding data entries can be created. So after the resources are created then we read the corresponding configuration of the resource and load it in memory in order to make process efficient.

Resource Record Fields
resourceId
source
target
category
gender
subCategory
createdAt
modifiedAt
createdBy
modifiedBy
description
verfiedFlag

Table 2: Resource Record Details.

4.3.3 Technologies used

For the controller part, for complete in-memory based architecture and for creating the microservices we used Java servlets with Apache-Tomcat-9 web server.

For the storage of the resources we used MongoDB-3.6.

For UI portion of the management system HTML5 along with CSS and Ajax have been used.

4.3.4 Features:

This resource management system has been integrated with two translation workbenches Transzaar (Ahmad et al., 2018). The features of the resource management system are given as follows:

- Addition of the resources in single and bulk mode.
- Editing and updating of the resources and their corresponding meta data.
- Provision to export and reuse the resources in various tools.
- Search: N-gram dynamic search mechanism.
- Easy import the resource in a translation workbench.
- Provision to develop and manage resources with linguistic features.
- Provision to verify them with a senior language expert of the corresponding language.

4.3.5 Process Flow

By using Kunji, translators can get aid in their translation tasks while using translation workbench. We demonstrated it with facilitating the translation workbench transzaar with Kunji. In kunji, they can create or import and use various kinds of resources like terminologies, bilingual and glossary dictionaries. Consider a scenario where a translator working on the workbench has some domain terminology dictionaries and wants to get help in his/her translation tasks. Then he can simply take that dictionary text file with source and target tab separated in each line in it. These resources can be imported/uploaded in the Kunji with the user metadata i.e. username, domain, language pair, and project etc.. So when translator opens his translation tasks and do editing then he can get the appropriate meanings of the terms or words of source text from terminology dictionary uploaded in Kunji. Those words are highlighted in source pane and on clicking them we can get appropriate meanings of the terms. Kunji facilitates a user working on workbench to add, import, export, update, n-gram search and edit the resources.

Possible types of the resources which can be uploaded are terminologies, glossaries, domain and bilingual dictionaries etc. Kunji also facilitates translator to import and use linguistic resources namely lexemes, named entities(NE), multiword dictionaries which can be helpful in selecting the correct form of the meaning of a word they should use.

4.4 Benefits of Kunji

It helps the translators by allowing them to search and look-up into their corresponding user specific or system specific resources, terminologies, glossaries which they can link with their translation tasks of a project to increase their productivity. Repetitive tasks are also facilitated by search or look up of a term which repeats many times in the task from the dictionaries and hence their consistency for given translation task would not be affected.Translators can import and use the linguistic resources like lexeme, NE and MWE dictionaries which can be helpful to decrease their cognitive load like selecting the correct form of the meaning of a word they should use. An experiment has been performed and is explained in Section-5 which would show how the linguistics resources can be beneficial for translators when we use this resource management system with a translation workbench.

Kunji also provides various aspects for managing and developing resources for MT application development. For NER and MWE, it provides provisions of category and subcategories like person/location/organization or Idiom/multiword phrases/Domain terms etc. Similarly, it also provides the ways to create different types of domain dictionaries according to the need. So the development process of MT can be hugely facilitated by using this NLP researchers-linguists, annotators and and NLP module developers can be benefited. It addresses issues like encodings, fonts, usage of the common resources and lexical dictionaries for multiple tools, import and export etc so using such tool can enhance their productivity.

5 Experiment

The described resource management system - Kunji has been integrated in translation workbenches Transzaar (Ahmad et al., 2018). We study the impact of providing this resource management system in CAT tool-Transzaar along with linguistic resources on the productivity of translators for English-Hindi language pair. The various types of resources namely terminologies, glossaries, TM along with linguistic resources like lexeme, NE and MWE dictionaries were made available in the resources management tool in the workbench.

5.1 Experimental setup

We setup the experiment of translation tasks from the news data of "The Hindu" news website [4].

5.1.1 Data Set

For the experiment, 20 news stories from The Hindu news paper From national domain have been taken. The data stories have been divided into 4 sets with each set containing 5 stories. The details of the data sets are given in Table 3 and Table 4.

We integrated the Kunji, the resource management system to the Tanszaar and integrated the resources namely terminologies, glossaries, TM along with linguistic resources like lexeme, NE and MWE dictionaries were made available in the resources management tool in the workbench. We divided the experiment of translation tasks into 6 scenarios which are described in Table 5.

[4] https://www.thehindu.com

Set No	Paras	Words	Sentences	Avg words per para	Avg paras per story	Avg sentences per para	Avg sentences per story
Set1(5 stories)	42	1180	65	28	8.5	1.54	13
Set2(5 stories)	64	1722	87	26.9	12.8	1.36	17.4
Set3(5 stories)	59	1553	83	26	11.8	1.4	16.6
Set4(5 stories)	54	1238	74	23	10.8	1.37	14.8

Table 3: Data sets description-1

Total words in dataset	5693
Total sentences in dataset	309
Total paras in dataset	219
Average no. of sentences in each data set	15.45
Average sentence length	18.34 tokens
Average paras in complete data set	11
Average para length	26 tokens

Table 4: Data sets description-2

Scenarios	Description
Scenario-1	Manual Translation
Scenario-2	Post-editing in GT on Text editor
Scenario-3	GT with Transzaar without Kunji
Scenario-4	GT with Transzaar and Kunji (onlyTM, Terms, Glossaries)
Scenario-5	GT with Transzaar and Kunji (only linguistic resources)
Scenario-6	GT with Transzaar and Kunji (TM, Terms, Glossaries and linguistic resources)

Table 5: Description of Various scenarios : Here GT refers to Google translation output

We have chosen the English-Hindi language pair for our experiment and created the translation tasks from the dataset for each of the mentioned scenarios in Table 5 and assigned them to the four translators. The translation tasks have been assigned in such a way that no translator would repeat the same set again in a given scenario.

6 Results and Discussion

The results of experiments are given in Table 6 and Table 7. Each cell in the Table 6 presents the total time taken(in hours) by corresponding translator in the post-editing of the each set. The cells in the last row present the average time taken on all sets in the corresponding scenario. Table 7 presents per sentence time taken by each translator(in minutes) in the 6 different scenarios given in Table 5.

From Table 6 and Table 7 we see the impact of a translation workbench with and without the availability of different kinds of resources. We observed the translation data with professional translators in Table 6 and Table 7. Scenario-3 is when we use a workbench without any resources. Compared to Scenario-3, in Scenario-4 we use the workbench with translation resources but exclude linguistic resources. We observe that the productivity is significantly improved by 29.93%. When we contrast this with Scenario-5 where only linguistic resources are used then the productivity is only improved by 17.12% over Scenario-3. When we use all the resources including linguistic resources then the productivity is improved by 45.54% as compared to Scenario-3. Hence, we see that the productivity is improved additionally by 15.61% over Scenario-4 when the resources with linguistic features are made available for the translators in the workbench along with TM, Terms and Glossaries. From the above mentioned results, it is evident that the translators' productivity is positively affected when they get help from the utilization of resources with linguistic features in CAT tool.

The scenario-wise time consumed per sentence translation by each of the translators is presented in Table 7. It is observed that the productivity of each translator increases when we go from Scenario-1 to Scenario-6. For each translator, it is observed that their productivity increases as they use the translation workbench with resources. Their productivity significantly increase by more than 66% in scenario-6 i.e. when the linguistic resources are made available for the translators in the workbench along with TM, Terms and Glossaries. In scenario-6, the corresponding productivity of the translators are increased by 66%, 71.6%, 71.3% and 68.2% respectively. We also see the significant improvement in productivity when we go from scenario-4(workbench with translation resources but exclude linguistic resources) to scenario-6((workbench with translation resources but including linguistic resources).

There are some observations of the translators about the data set. The Set-1 is the toughest of all four sets. It is evident from table-7 that, translators took most time in translation of this set in each of the scenario. The Set-4 is the easiest of the sets as less time is consumed in its translation. Translators found the linguistic dictionaries (lex-

Set No	Scenario-1 (Total Time in hours)	Scenario-2 (Total Time in hours)	Scenario-3 (Total Time in hours)	Scenario-4 (Total Time in hours)	Scenario-5 (Total Time in hours)	Scenario-6 (Total Time in hours)
Set1	1.986(U1)	1.211(U2)	1.16(U3)	0.826(U2)	0.96(U4)	0.702(U21)
Set2	1.733(U4)	1.12(U1)	1.064(U2)	0.734(U1)	0.858(U3)	0.563(U4)
Set3	1.464(U3)	0.8431(U4)	0.835(U1)	0.5925(U4)	0.6864(U2)	0.421(U3)
Set4	1.158(U2)	0.702(U3)	0.64(U4)	0.443(U3)	0.565(U1)	0.33(U2)
Avg Time Taken in the scenario	1.585	0.97	0.9236	0.647	0.766	0.503

Table 6: Results of Experiment

Scenarios	Total Time taken by U1 per sent. (in min.)	Total Time taken by U2 per sent. (in min.)	Total Time taken by U3 per sent. (in min.)	Total Time taken by U4 per sent. (in min.)
Scenario-1	1.84(S1)	0.938(S4)	1.058(S3)	1.195(S2)
Scenario-2	0.78(S2)	1.03(S1)	0.569(S4)	0.61(S3)
Scenario-3	0.6(S3)	0.733(S2)	1.07(S1)	0.52(S4)
Scenario-4	0.506(S2)	0.76(S1)	0.359(S4)	0.43(S3)
Scenario-5	0.41(S4)	0.49(S3)	0.59(S2)	0.88(S1)
Scenario-6	0.64(S1)	0.267(S4)	0.304(S3)	0.38(S2)
Improvement (in %)	66%	71.6%	71.3%	68.2%

Table 7: Scenario-wise Results of Experiment with Professional Translators

eme, named entities and MWE) very helpful in the translation. For example the meaning of the word "dreaded" contains two senses in different contexts in Hindi. The lexeme dictionary aided the translators in disambiguation of word senses.

7 Conclusion

We have proposed a resource management system which addresses the issues of the translators as well as MT system development. Using Kunji, a translator can utilize, reuse and manage his resources which can be shared across projects and can be used further. Also the NLP researchers get benefited using it in their resources management in MT system development so that the resources can be utilized and managed in an efficient way.

It also provides the mechanism which allows a translator to get help from the linguistic resources-lexeme, NER and MWE dictionaries in addition to their regular translation memories, concordances and terminologies. We performed experiments which show the effect of the linguistic resources in the productivity of the translators. We see that their productivity increased by 15.61% when they use linguistic resources along with their regular TM, terminologies and glossaries as mentioned in Table 6 and Table 7. We hope to extend this system by analyzing the impact of more complex NLP module like Morphological analyzer in the translation pipeline.

References

Rashid Ahmad, Priyank Gupta, Nagaraju Vuppala, Sanket Kumar Pathak, Ashutosh Kumar, Gagan Soni, Sravan Kumar, Manish Shrivastava, Avinash K Singh, Arbind K Gangwar, et al. 2018. Transzaar: Empowers human translators. pages 1–8.

Vicent Alabau, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Ulrich Germann, Jesús González-Rubio, Robin Hill, Philipp Koehn, Luis Leiva, et al. 2014. Casmacat: A computer-assisted translation workbench. pages 25–28.

Wilker Aziz, Sheila Castilho, and Lucia Specia. 2012. Pet: a tool for post-editing and assessing machine translation. pages 3982–3987.

Sriram Chaudhury, Ankitha Rao, and Dipti M Sharma. 2010. Anusaaraka: An expert system based machine translation system. pages 1–6.

Hamish Cunningham. 2002. Gate, a general architec-

ture for text engineering. *Computers and the Humanities*, 36(2):223–254.

Marin Dimitrov, Hamish Cunningham, Ian Roberts, Petar Kostov, Alex Simov, Philippe Rigaux, and Helen Lippell. 2014. Annomarket–multilingual text analytics at scale on the cloud. pages 315–319.

Marcello Federico, Nicola Bertoldi, Mauro Cettolo, Matteo Negri, Marco Turchi, Marco Trombetti, Alessandro Cattelan, Antonio Farina, Domenico Lupinetti, Andrea Martines, et al. 2014. The matecat tool. pages 129–132.

Nancy Ide and Laurent Romary. 2009. Standards for language resources. *arXiv preprint arXiv:0909.2719*.

GROUP ILMT. 2009. Sampark: Machine translation system among indian languages. http://sampark.iiit.ac.in.

Rafał Jaworski. 2013. Anubis-speeding up computer-aided translation. pages 263–280.

Rajeev Sangal and Dipti Misra Sharma. 2001. Creating language resources for nlp in indian languages 1. background.

Anil Kumar Singh. 2008. A mechanism to provide language-encoding support and an nlp friendly editor.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. pages 102–107.

Johannes Thönes. 2015. Microservices. *IEEE software*, 32(1):116–116.

Identification of Synthetic Sentence in Bengali News using Hybrid Approach

Soma Das
soma_phd_2018july@iiitkalyani.ac.in
Indian Institute of Information Technology Kalyani,
West Bengal, India

Sanjay Chatterji
sanjayc@iiitkalyani.ac.in
Indian Institute of Information Technology Kalyani,
West Bengal, India

Abstract

Often sentences of correct news are either made biased towards a particular person or a group of persons or parties or maybe distorted to add some sentiment or importance in it. Engaged readers often are not able to extract the inherent meaning of such synthetic sentences. In Bengali, the news contents of the synthetic sentences are presented in such a rich way that it usually becomes difficult to identify the synthetic part of it. We have used machine learning algorithms to classify Bengali news sentences into synthetic and legitimate and then used some rule-based postprocessing on each of these models. Finally, we have developed a voting based combination of these models to build a hybrid model for Bengali synthetic sentence identification. This is a new task and therefore we could not compare it with any existing work in the field. Identification of such types of sentences may be used to improve the performance of identifying fake news and satire news. Thus, identifying molecular level biasness in news articles.

Keywords: Synthetic Sentence, Engaged Reader, Machine Learning technique, Rule Base Approach

1 Introduction

The Bengali language is rich in terms of the usage of its words. It is also a relatively free word order language. By changing the order of the same set of words, the author can add some emphasis to some part of the sentence. It is usually observed that the Bengali sentences are frequently distorted like this way. The number of ways English sentences can be distorted is much less than the number of ways a Bengali sentence can be. But all the distorted sentences not necessarily have added biasness or emphasis.

Some readers take the inherent meaning of the sentences without getting into involved in the biased part of it. They can take out an overview of the text. But often, an engaged reader gets engaged with the writer's views. Sometimes it is not so harmful or it is preferred to be an engaged reader. For example, to get the full flavour of a literary work, the reader has to be engaged. But often it is not desirable. For example, in a piece of political news, it is not recommended to engage a reader without his concern. So, it is essential to notify the reader about synthetic sentences.

Often sentences of correct news are either made biased towards a particular person or a group of persons or parties or maybe distorted to add some sentiment or importance in it. We refer such types of sentences as synthetic sentences. Engaged readers often are not able to extract the inherent meaning of synthetic sentences. In Bengali, the news contents of the synthetic sentences are presented in such a rich way that it usually becomes difficult to identify the synthetic part of it.

In this paper, we wish to identify the synthetic sentences in Bengali news automatically. We use the linguistic features in multiple Binary Machine Learning Classifiers to decide whether it is synthetic or legitimate. Then analyzing a confusion matrix, we apply a set of rules. Finally, we combine these models using a voting-based approach. We test this hybrid technique in a Bengali news corpus covering the news in Politics, Sports, Entertainment, and Social domains. The final hybrid technique is able to provide 86% accuracy.

The rest of the paper is organized as follows. Section 2 illustrates a background study related to synthetic news detection. Section 3 and Section 4 discuss how we have prepared the experimental dataset and model building part. Section 5 shows the results of different steps. Finally, Section 6 presents concluding remarks of the task.

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 193–200
Hyderabad, India, December 2019.

2 Related Work

There are some works in the detection of fake news. They detect fake news in a news corpus Bovet and Makse (2019); Batchelor (2017); Shu et al. (2017); Conroy et al. (2015) that is, misleading news stories which come from non-reputable sources. These papers mainly focus on fake news from four perspectives: the false knowledge, its writing style, its propagation patterns, and the credibility of its creators and spreaders.

Rubin et al. (2016) describes three types of fake news in contrast to reporting. These are - serious fabrications (uncovered in mainstream or participant media); large-scale hoaxes; humorous fakes (news satire, parody).

Zellers et al. (2019) discussed the threats posed by automatically generated propaganda articles that closely mimics the style of real news. They have designed a language model-based system called Grover for the controllable generation of text from the title of the news. Humans may find this generated text to be more trustworthy compared to the actual news article. Such type of fake news called neural fake news is discriminated best using the generator system itself.

Bradshaw and Howard (2017) compared the teams who spread manipulated information, also called disinformation through social media and news across 28 countries including India. These types of fake news are created manually to influence the voters and domestic audiences purposely. Melford and Fagan (2019) designs a Global Disinformation Index (GDI) to combat the disinformation.

Another essential type of fake news is created by proliferating stylistic bias in the text. Pérez-Rosas et al. (2017) have used linguistic features in Support Vector Machine (SVM) to detect these fake news in some English newspapers. Rubin et al. (2016) discriminated between synthetic and legitimate news using 5 features namely Absurdity, Humor, Grammar, Negative Affect, and Punctuation.

3 Dataset Preparation

We evaluate our proposed framework on two datasets, Kaggle Bengali news, and Online Bengali news. For the time being, we are not using the name of the newspapers to avoid the controversy. In total, we have 25K news covering seven different domains, namely Kolkata, State, National, Sports, Entertainment, World, and Travel. Each news contains on an average of 15 sentences. The distribution of the Bengali news dataset in the seven domains is shown in Fig. 1

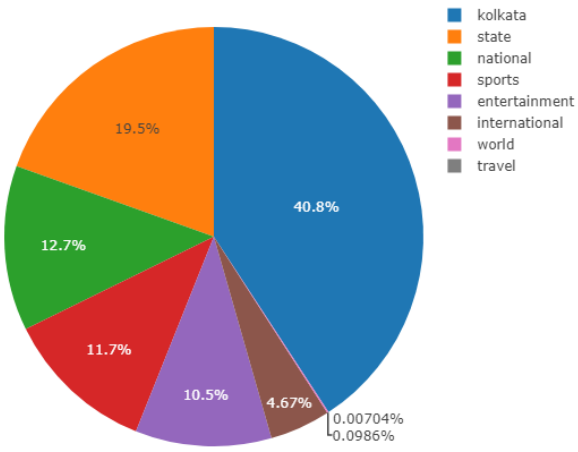


Figure 1: Bengali News Dataset Distribution in Kolkata, State, National, Sports, Entertainment, World, and Travel Domains

3.1 News Content Features

The structure of the news dataset is listed below:

1. **Source:** Author or publisher of the news article.
2. **Domain:** The domain of the news is defined in this field. In this dataset, we have seven different domains viz, Kolkata, State, National, Sports, Entertainment, World, and Travel.
3. **Headline:** Short title text that aims to catch the attention of readers and describes the main topic of the article
4. **Body Text:** Main text that elaborates the details of the news story, there is usually a significant claim that shapes the angle of the publisher.

Depending on these raw content attributes, different kinds of feature representations can be built to extract discriminative characteristics of synthesis news. The news content we are looking at mostly be linguistic-based features, discussed in the following section.

3.2 Preprocessing of News Sentences

The overall framework of the machine learning-based classifier is divided into three parts: Cleaning of raw text, feature extraction and synthetic

news classification. The collected news sentences are annotated manually as a legitimate or synthetic sentence. It is difficult to deal with raw news due to noise. The noisy news includes:

- Keyphrases: - নিজস্ব প্রতিবেদন, ওয়েব ডেস্ক:, এই বিষয়ে অন্যান্য খবর, ব্যুরো, ডিজিটাল ডেস্ক, সূত্রের খবর
  Different publication media use the mentioned key phrases which are actually not part of the news. We remove these phrases from the news sentences.

- English Sentences: News contains some English sentences along with Bengali sentences. The following English sentence is highlighted inside a Bengali sports news, e.g.,*"Delhiites get a bite of #ViratKohli quite literally at #MadameTussauds PC Statesman pic.twitter.com/FNLARdIQi6 - Bharat Sharma (sharmabharat45) June 7, 2018", "ISIS," "JNU"*. We remove such non Bengali sentences from our dataset.

- Stop Words: Stop words are described as the most common words that occur in any corpus of a particular language. At the preprocessing step, we remove stop words such as - 'এ', 'এবং', 'আর' from the sentences. Here we created a stop word list of 360 words and then these words are removed from the corpus.

- Word Stemming: Word stemming is applied to map the words with different endings to a single one such as চ্যালেঞ্জের,চ্যালেঞ্জ becomes চ্যালেঞ্জ. Bengali is a very inflectional language for which stemming is required for further processing.

- Other: News contains emoticons, symbols, and pictographs. We remove them by using Unicode.

By using the above-mentioned list of phrases, we preprocess the raw news and generate a clean text for further processing.

3.3 Annotation Guideline

We have annotated Bengali news sentences into two categories: synthetic and legitimate. In this section, we discuss the method we have followed in tagging with some examples.

- *Example-1:* এই সম্প্রদায়ের এক সদস্যের মতে, ২০১৫ সালে আমেরিকাতে এবং ২০১৭ সালে ইংল্যান্ডে সভা করেছিলেন প্রধানমন্ত্রী।
  [According to a member of this community, the Prime Minister had a meeting in the United States in 2015 and England in 2017.]
  In this sentence, it is claimed that the statement is taken from somebody, but the name is not mentioned explicitly. This is why, we consider such types of sentences as synthetic. If the name of the claimer is added, then it is converted to legitimate.

- *Example-2:* সুপার ওভারেও খেলার শেষ না হওয়ায় বাউন্ডারির সংখ্যার ভিত্তিতে ইন্ডিয়াকে চ্যাম্পিয়ন ঘোষণা করে দেওয়া হয়।
  [India was declared champion on the basis of the number of boundaries as the game did not end in the Super Over.]
  The cause-effect sentence of Example-2 is considered legitimate as it is based on a true fact cause, and the relation is an established relation: if a game does not end in Super Over then go for a number of boundaries.

- *Example-3:* সুপার ওভারে উত্তেজনা, শিষ্য নিশামের ছক্কা দেখে শেষ নিঃশ্বাস গুরুর।
  [Tension in the Super Over, the master releases last exhale after seeing the six of the disciple Neesham.]
  In the cause-effect sentence of Example-3, the cause is a true fact, but the relationship is based on an assumption or probability. There is no rule in the environment to state that one will die after seeing one's six. Therefore this sentence is tagged as synthetic.

- *Example-4:* সম্ভবত মন খারাপের কারনে, বোল্ট তার সেরা পারফরম্যান্স দিতে পারলনা।
  [Probably due to distress, Bolt could not give his best performance.]
  This is also a cause-effect sentence. In this sentence, the cause is not a true fact as a probability is associated with it. Though, the relation "if somebody is in distress, then he will not be able to give his best performance" is an established relation this sentence has synthetic property.

- *Example-5:* The phrases containing synthetic adjectives have a synthetic property like: মন ভাঙা বোল্ট বললেন

[broken heart Bolt told]
We observed that verbs carry the best clue about the synthetic property. Therefore we have created a clue verb list and used it as a feature. But there are some adjectives, adverbs, nouns, which can also be considered a clue. We considered them during annotation.

- *Example-6:* কিন্তু, আমরা তাঁদের আশাপূরণ করতে পারিনি।
[But we could not meet their hope.]
This sentence talks about an abstract mental state (hope). It is not defined how to measure whether it is met or not. Therefore, this is a synthetic sentence.

- *Example-7:* তবু মণীশ পান্ডের শতরানের সৌজন্যে বড় রান তুলে ফেলে ভারতীয়।
[Yet by courtesy of Centurion Manish Pandey India built a big score.]
Here, the word 'yet' makes this sentence synthetic, as it means it would not be possible without him. But it is not correct as others are not tested. Dropping this word leads to a legitimate sentence.

4 Proposed System

After preprocessing of the raw news, the news is tokenized and segmented into sentences level. In this paper, we create a hybrid approach by combining Binary Machine Learning Classifiers using the Voting approach and then postprocessing with a Rule-Based approach. The proposed system is as follows:

1. For machine learning, features are generated from the sentences, and after that, we apply supervised machine learning algorithms, namely Support Vector Machine, Naive Bayes, K Nearest Neighbors, Random Forest, Decision Tree, and Logistic Regression.

2. According to the results of supervised algorithms, we are creating some rules based on the mismatched outputs.

3. Lastly, we are combining the supervised algorithms using a voting approach. We are giving a higher preference for synthetic tagging. If among the six classifiers, 3 classifiers tell the sentence is synthetic and 3 classifiers tell it is legitimate, then we annotate it as synthetic. However, if more than 3 classifiers tell that the sentence is legitimate, then it became legitimate.

In this paper, we consider synthetic news classification for independent sentences as each sentence carries some synthetic or legitimate property. Our approach is to use a committee of classifiers, each trained on a set of text features. The entire list of features is presented in this section.

4.1 Feature Selection in Synthetic Classification

Feature Selection of any classification problem takes a crucial part. Each sentence is represented by a feature vector which contains numerical features that represent the occurrence frequency or weight of a feature or binary features (occurrence or non-occurrence of a feature) or a ternary feature. The features are listed below:

1. Punctuation: Various punctuation is used to indicate different types of sentiments. The use of punctuation can help the synthetic news detection algorithm to differentiate between funny, entertaining, deceptive, and truthful texts. This feature has two values (binary) - Exclamation (!) and Question Mark (?). This feature is used because, according to our observation the exclamatory sentences and question sentences are more prone to be synthetic. This feature helped us to improve the precision of synthetic sentence identification.

2. Named Entities: News tells a story related to a particular incidence. Legitimate news is having the property of some person telling something or making comments. Named Entities are used many times inside news. We have observed that the sentences having a Named Entity mostly become a legitimate sentence. This is because even if an incidence is false or synthetic, but it is told by a particular person we consider it to be true. Consider the example: "a person x told that he is probably sick". Here the 'probably' word makes the statement synthetic. But the sentence is legitimate as the person x has actually told it. Thus, use this binary feature (there exist a Named Entity or not) may be helpful to predict synthetic news sentence.

   To find the named entity in the news sentence, we manually annotate 42k words from 6k

news sentences to train CRF++ model. Here, we are using POS tag as a feature of CRF++ to find the presence of named entity in the sentence. To get the POS tag we have used some of the features proposed by Dandapat et al. 2007 which are available with us. Along with the POS tag the Bengali gazetteer list of 1200 names is also used in training the CRF++ model. Then the model is tested with our 530 news examples and we get 94.3% accuracy. We have used this Bengali gazetteer list as a binary feature (presence or non-presence of named entity).

3. Domain-specific Clue Verb: We have seen that maximum synthetic sentences have some verbs which carry some indication for the sentence to have synthetic property. Similarly, some verbs are indications of the legitimate property of the sentence. Theses verbs are defined as Clue Verbs. We are attempting to make a list of words for Legitimate sentences and another for synthetic sentences manually. By analyzing a large corpus, we make a list as shown in 1, which is not exhaustive.

   We have considered this clue verbs as a ternary feature as follows. If there is a Legitimate Clue Verb then it is 1 (we do not need to consider Synthetic Clue Verb); if there is no Legitimate Clue Verb, but there is a Synthetic Clue Verb then it is 2, and if there is neither Legitimate Clue Verb nor Synthetic Clue Verb then it is 3. The value 1 indicates that the sentence is a strong candidate for being legitimate; the value 2 indicates that the sentence is a strong candidate for being synthetic and the value 3 indicates that there is no clue about its property. According to our observation, this feature is the most effecting feature in the classification task.

4. TF-IDF: Term Frequency - Inverse Document Frequency (TF-IDF) of a term is used to denote its importance. TF(w,d) denotes the raw count of the word (w) in a news document (d) and IDF(w,D) is a measure of how much information the word (w) provides, i.e., if it is usual or rare across all news documents (D). Finally, TF-IDF is defined as follows.

$$tf\_idf(w, d, D) = tf(w, d) \times idf(w, D) \quad (1)$$

   Thus, TF-IDF is used to determine the importance of words in news domain. We have combined the TF-IDF of words of the input sentence to calculate the importance of sentence in news domain. Considering that the synthetic sentences carry more importance we have used it as a feature in our task.

4.2 Machine Learning-Based Classification

Synthetic news sentence classification may be done at the document level, sentence level, and phrase level. We are considering the news article is based on an actual fact. Some of its sentences are synthesized by the author to attract the engaged reader. Our objective is to identify those synthetic sentences. So, in this paper, sentence-level classification is considered where an independent sentence is classified as synthetic sentences and legitimate sentences.

A supervised binary classifier algorithm may be used to identify the synthetic sentences. Several supervised machine learning techniques have been examined in the paper to classify the sentences into classes. Those are Support Vector Machine, Naive Bayes, K Nearest Neighbors, Random Forest, Decision Tree, and Logistic Regression (LR). We have considered the Punctuation feature, Named Entity feature, clue verb, and TF-IDF feature, as discussed in Section 4.1.

The Confusion Matrix is one of the most intuitive metrics used for finding the correctness and accuracy of the model. The Confusion Matrix in itself is not a performance measure as such, but almost all of the performance metrics are based on the Confusion Matrix and the numbers inside it. The confusion matrix is a table with two dimensions (“Actual” and “Predicted”), and sets of ”classes” in both dimensions.

The following terms are associated with a confusion matrix.

- True Positive (TP): when predicted synthetic sentences pieces are actually annotated as a synthetic sentence;
- True Negative (TN): when predicted legitimate sentences pieces are actually annotated as true sentences;

Label	Clue Verb
Legitimate	বলছেন [Says-H], জানিয়েছেন [Said-H],জানান [Said-H], বললেন [Said-H],ঘোষেণামতো [Declaration], কথা বলেন[Speak-H]কথা বললেন [Speak-H], জানান[Tell me], জানালেন[Told], ঘোষণা করলেন[Announced]
Synthetic	খতিয়ে দেখা[Check it out], খতিয়ে দেখেন[Check it out],খতযি়ে দেখা হয়[Is checked], প্রশ্ন করেছেন [Asked-H], প্রশ্ন করে [Asked-H], প্রশ্ন করা হয়েছে[Asked-H], প্রশ্ন করা[Asked-H], প্রশ্ন হয়েছে [The question has been], প্রশ্ন উঠেছে [The question arises], কথা ছিল [There was talk], কথা দেওয়া[Promise-H], কথা দিয়েছিল[Promised-H], কথা দিয়েছিলেন[Promised-H],কথা দিয়ে রাখেননি[Didn't talk], প্রশ্ন উঠতে শুরু করেছে[The question has started to arise], আস্থা নেই[Not confident], মনে করা হচ্ছে[It seems], চ্যালেঞ্জ নিয়েছেন [Have taken up the challenge], চ্যালেঞ্জ ছোড়েন[Throw the challenge], চ্যালেঞ্জ নেওয়া [Take up the challenge], চ্যালেঞ্জ ছোড়া[Throw the challenge], চ্যালেঞ্জ[challenge-H],মনে করা হচ্ছে[It seems], প্রমাণিত হবে[Will prove], নীতি নিয়েছে[Policy taken], সক্রিয় ভাবে পথে নামতে দেখা গিয়েছে[Actively shown on the way down], দাবি <Null verb>[Claim-H], খবর চাউর হয়ে যায়[The news goes sour], আতঙ্কের ছাপ <Null Verb>[The impression of terror], আতঙ্কের ছাপ নেই[No sign of panic], তদারকি করেন[Take care], ফের সতর্ক করে দেন[Warns again], দাবি তুলেছিলেন[Claimed-H], তবু স্বস্তি ছিল[Yet there was relief], দেখছেন স্থানীয়রা[The locals are watching], আস্থা নেই[Not confident], আশ্বাস দিয়েছেন[Assured-H], আশঙ্কা[Fear], আশঙ্কা করেছেন[Have feared], আশঙ্কা করা[Do not be afraid], আশঙ্কা করল[Apprehensive], রয়েছে বলে[Say there is], দেওয়ার অভিযোগ[The charge to give], চাঙ্গা হয়ে ওঠে[Became stronger], অভিযোগ তুললেন[Complain-H], অভিযোগ করল[Complained-H], অভিযোগ করা হয়েছে[The complaint was made], অভিযোগ[Complain-H], কড়া বার্তা[Strong message], রুখে দাঁড়িয়েছিলেন [Standing in the stands], রুখে দাড়ান-H[Stand up], রুখে দাড়ালেন-H[Stand up], তোপের মুখে-H[Under the cannon], মাঠে নামাচ্ছেন-H[Getting down on the field], স্বীকৃতি দিয়েছেন[Recognized], স্বীকৃতি দেওয়া[recognition-H], স্বীকৃতি মিললে[Acceptance-H], বিচারের মুখোমুখি[Facing trial], অভিযোগ উঠল[The complaint arose], অভিযোগ উঠা[Complaints-H arise], বলে অভিযোগ[Complain-H], একাংশের দাবি ছাপ <Null Verb>[Part claim impression], একাংশের মত[Like a part], উস্কে দিল[Instigated-H], কীসের ইঙ্গিত[What a hint], জানা গিয়েছে[Got it]

Table 1: List of Clue Verb [H: Honorific]

- False Negative (FN): when predicted legitimate sentences pieces are actually annotated as synthetic sentences;
- False Positive (FP): when predicted synthetic sentences, pieces are actually annotated as legitimate sentences.

We have created a separate set of 106 sentences to create the Confusion Matrix. The Confusion Matrices of all the Binary Classifier techniques for these sentences are shown in Fig. 2.

4.3 Rule Based Postprocessing

After analyzing the errors in the confusion matrix of the Binary Classifier techniques, as shown in Fig. 2, we have formulated an initial set of rules. These rules are used in the postprocessing step to correct some of the errors. The rules we formulated are discussed below.

1. If the **topic** of the news sentence is clubbed with old news of different **topic**, then it is considered as synthetic. Consider the following example.
   সারদা মামলায় রমেশ গাঁধীকে নিজেদের হেফাজতে নেওয়ার পরে এ বার রোজ ভ্যালির দুই কর্তাকে গ্রেফতার করল তারা|
   [After taking Ramesh Gandhi in their custody in Sarada case now they arrested two heads of Rose Valley]
   In this sentence, the leading news on arresting two heads of Rose Valley is clubbed with old news of different topics. Therefore it is considered synthetic.

2. If in the news sentence the reason for an incidence is written abstractly, then it is considered as synthetic. Consider the following example.
   শাসকদলের গোষ্ঠীকোন্দলের জন্য সোমবার চাষিদের নথিপত্র জমা দেওয়ার শিবির বেড়াবেড়ি থেকে সরে গেল সিঙ্গুর বিডিও অফিসে|
   [Due to infighting in the governing party Monday the camp for submitting documents of farmers is moved from Beraberi to Singur BDO office.]
   We have prepared an initial list of abstract reasons. In this sentence, the reason ”infighting” is not concrete. Therefore it is considered synthetic.

3. If there is an incomplete list of names, matters, topics, or any other thing in the sentence, then it is considered as synthetic. Consider the following example.

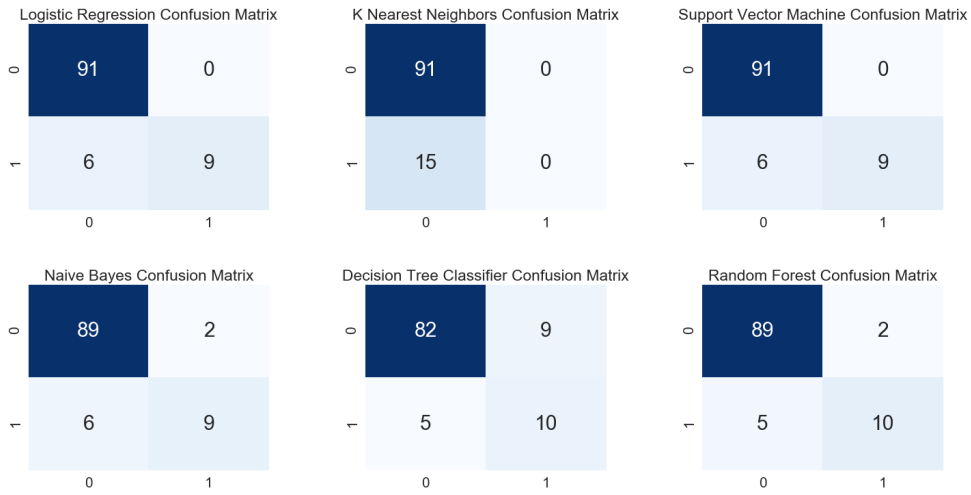


Figure 2: Confusion Matrix after Applying 6 Binary Classifiers on 106 Sentences[X axis denotes predicted value and Y axis denotes actual value and both cases 0 indicates legitimate sentence and 1 indicates Synthetic Sentence]

এজন্য বর্ধমান, কল্যাণী এবং অন্যান্য কিছু বিশ্ববিদ্যালয়কে আবেদন জানাতে বললেন শিক্ষামন্ত্রী|
[For this, the education minister asked the Bardhaman, Kalyani and some other universities to apply.]
In this sentence, the phrase ”some other universities” indicates that an incomplete list is used. Therefore it is considered synthetic.

4. If a sentence contains a legitimate clue verb, then it is classified as a legitimate sentence. But with the clue verb if there exists an adverb, then the sentence becomes synthetic. Consider the following sentence.
শিক্ষার সর্বস্তরে শিক্ষকশিক্ষিকাদের হাজিরায় বিশেষ নজর দেওয়ার কথা তিনি বারবার বলেছেন
[He has repeatedly said to pay special attention to the attendance of teachers in all levels of education.]
In these sentences, an adverb is used with the clue verb. Therefore it is considered as synthetic.

5. If in the sentence there exist any phrase in Double Quotation indicating a comment, then it is a legitimate sentence irrespective of the property of the comment. Consider the following example.
তিনি জানান, "এই বিষয়ে সকলের জন্য একটি সাধারণ নিয়মাবলী থাকলে ভাল হয়"
[He said, ”it is better to have a general rule for everyone in this regard”.]
This sentence tells the comment made by some entity. Therefore it is considered legitimate.

5 Experimental Result

Various evaluation metrics have been used to evaluate the performance of machine learning models. We want to test the performances of our models in terms of Accuracy, Precision, Recall, and F1-Score. These metrics are commonly used to evaluate the machine learning models and enable us to evaluate the performance of a classifier from different perspectives. The results of the k-fold cross-validations for each of our hybrid models are shown in section 5.1.

Then we have applied the voting approach to combine the models and then the rules. The final accuracy of this hybrid system is discussed in section 5.2.

5.1 Classification Performance of Individual Hybrid Machine Learning Models

Firstly, we have tested the six binary classifiers namely Logistic Regression, K Nearest Neighbors, Support Vector Machine, Naive Bayes, Decision Tree, and Random Forest. We have used 530 sentences annotated as legitimate or synthetic. These sentences are folded into Training and Test data in the proportion of 80:20.

We have used k-fold cross-validation ($k = 5$) and calculated the above metrics in each of the k-folds for each model. Then, we have manually applied the rules on all the six prediction models. We prepared a comparison chart of the final performances of these six individual hybrid systems which is given in Fig. 3.

In Fig. 3, the dark gray colour bars indicate the k-fold cross-validation results of the binary clas-

Approach	Accuracy	Precision	Recall	F1-Score
Logistic Regression based Hybrid System	0.82	0.67	0.82	0.74
Combined Model based Hybrid System	0.86	0.86	0.87	0.85

Table 2: Performances of different approaches for Synthetic Sentence classification

Figure 3: Classifier Result

sification. The light gray colour bars indicate the result we got after applying the rules on the best output of the corresponding technique. The result shows that the rules improved the Support Vector Machine and LR based techniques most. The Logistic Regression based technique combined with the rules gave the highest accuracy up to this stage. We are getting highest around 82% accuracy by applying rules on the Logistic Regression based model.

5.2 Classification Performance of Combined Hybrid Machine Learning Model

Finally, we have used a voting based combination of these six machine learning classifiers. If a sentence is tagged as synthetic by 3 or more classifiers then we consider it to be synthetic. Otherwise, it is considered to be legitimate. Then, we have applied the rules on the combined classifier. The final result of this hybrid system is shown in Table 2.

6 Conclusion and Future Scope

In this paper, we defined and compared synthetic and legitimate sentences and highlighted many interesting differences between these two categories. We then utilized these differences as features to detect synthetic sentences. We have proposed a hybrid approach that can detect synthetic news. To the best of our knowledge, our work is the first attempt to detect synthetic news at the Bengali news sentence level. In the future, we want to extend it to use semantic features in the Machine Learning model and calculate the degree of synthetic property in the synthetic sentences. Then we want to compare the renowned Bengali newspapers in terms of the usage of different types of synthetic sentences.

References

Oliver Batchelor. 2017. Getting out the truth: the role of libraries in the fight against fake news. *Reference services review*, 45(2):143–148.

Alexandre Bovet and Hernán A Makse. 2019. Influence of fake news in twitter during the 2016 us presidential election. *Nature communications*, 10(1):7.

Samantha Bradshaw and Philip Howard. 2017. Troops, trolls and troublemakers: A global inventory of organized social media manipulation. *Technical report, Oxford Internet Institute*.

Niall J Conroy, Victoria L Rubin, and Yimin Chen. 2015. Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Sandipan Dandapat, Sudeshna Sarkar, and Anupam Basu. 2007. Automatic part-of-speech tagging for bengali: An approach for morphologically rich languages in a poor resource scenario. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 221–224. Association for Computational Linguistics.

Clare Melford and Craig Fagan. 2019. Cutting the funding of disinformation: The ad-tech solution. *Technical report, The Global Disinformation Index*.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2017. Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*.

Victoria Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 7–17, San Diego, California. Association for Computational Linguistics.

Kai Shu, Suhang Wang, and Huan Liu. 2017. Exploiting tri-relationship for fake news detection. *arXiv preprint arXiv:1712.07709*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news.

An LSTM-Based Deep Learning Approach for Detecting Self-Deprecating Sarcasm in Textual Data

Ashraf Kamal
Department of Computer Science
Jamia Millia Islamia, (A Central University)
New Delhi, India
ashrafkamal.mca@gmail.com

Muhammad Abulaish
Department of Computer Science
South Asian University
New Delhi, India
abulaish@sau.ac.in

Abstract

Self-deprecating sarcasm is a special category of sarcasm, which is nowadays popular and useful for many real-life applications, such as brand endorsement, product campaign, digital marketing, and advertisement. The self-deprecating style of campaign and marketing strategy is mainly adopted to excel brand endorsement and product sales value. In this paper, we propose an LSTM-based deep learning approach for detecting self-deprecating sarcasm in textual data. To the best of our knowledge, there is no prior work related to self-deprecating sarcasm detection using deep learning techniques. Starting with a filtering step to identify self-referential tweets, the proposed approach adopts a deep learning model using LSTM for detecting self-deprecating sarcasm. The proposed approach is evaluated over three Twitter datasets and performs significantly better in terms of *precision*, *recall*, and *f-score*.

1 Introduction

Over a decade, the popularity of the micro-blogging platform, Twitter, has significantly increased for analyzing its content for varied real-world applications. The information extracted from Twitter can shed light on numerous applications, such as text categorization, sentiment analysis, election campaign and result prediction, open-source intelligence, and event detection. However, the contents available on Twitter in the form of tweets are short and limited to maximum 280 characters. Moreover, tweets are informal and mainly consist of misspelled words, slangs, bashes, acronyms, shortened words, non-literal unstructured phrases, and emoticons. Due to existence of such volumunious informal texts in the form of tweets text information processing is a challenging task. Moreover, analysis of the tweets has become more challenging due to presence of figurative language, especially sarcasm. The main role of a sarcastic tweet is to reverse the actual polarity and alter the literal semantics. However, the computational detection of sarcasm benefits many applications, especially opinion mining and sentiment analysis systems (Bouazizi and Ohtsuki, 2015).

The online Macmillan dictionary defines sarcasm[1] as "*the activity of saying or writing the opposite of what you mean, or of speaking in a way intended to make someone else feel stupid or show them that you are angry*". Sarcasm is the most seen figurative language category over online social media platforms. The presence of sarcasm in tweets is dramatically rising and computational detection of sarcasm is a challenging and interesting task. It is widely covered by researchers in recent years, but the study on different categories[2] of sarcasm, such as self-deprecating sarcasm, is very limited. Self-deprecating sarcasm[3] is a special category of sarcasm in which users mainly apply sarcasm over themselves using disparage, ridicule, and contemptuous remarks in a sarcastic style using humor. It is defined as a "sarcasm that plays off of an exaggerated sense of worthlessness and inferiority". For example, the phrase *love going to the office on Sunday* in the text "*Really, I always love going to the office on Sunday*" represents a self-deprecating sarcasm.

Nowadays, self-deprecating sarcasm has become a new style of product marketing and campaign strategy. It is mainly used for product endorsement purposes. This new marketing and campaign strategy is mainly used to excel the busi-

[1] https://bit.ly/2WsUkUk (last accessed on 15-Nov-19)

[2] https://literarydevices.net/sarcasm/ (last accessed on 15-Nov-19)

[3] https://bit.ly/2vwjtid (last accessed on 15-Nov-19)

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 201–210
Hyderabad, India, December 2019.

ness growth, but without losing the brand value (Kamal and Abulaish, 2019). The main aim of this strategy is to draw the attention of the customer towards the brand. As per the American marketing association[4], "self-deprecating advertising means consumers can see a different side to brands, making them more relatable and down-to-earth". Interestingly, after an in-depth analysis of tweets, we found that there are many tweets in which users refer themselves. We consider such tweets as *self-referential* or *self-deprecating*. For example, "*Really, I just love it*" is a self-referential tweet. Our analysis further reveals that some of the self-referential tweets are self-deprecating using sarcasm, i.e., in these tweets users undervalue, criticize, insult, and disparage themselves using sarcastic phrases. We consider all such self-referential tweets as self-deprecating sarcasm.

In this paper, we propose a deep learning approach using `Long Short-Term Memory` (LSTM) to detect self-deprecating sarcasm in textual data like tweets. Initially, after preprocessing, we first identify self-referential tweets from the dataset based on a set of patterns, and rest of tweets are filtered out. The main motivation behind the filtration of the non-self-referential tweets is to increases the overall efficiency of the self-deprecating sarcasm detection process. In brief, the main role of the self-referential tweets identification module can be summarized as follows:

- Identification of explicit self-referential tweets: After an in-depth analysis across all the datasets, we identify a set of patterns followed by the self-referential tweets. Table 1 presents a set of regular expression based patterns and it is categorized as *specific patterns* and *generic patterns*. The specific patterns are based on tags and tokens present in the tweet which indicate self-referential nature of the tweet. On the other hand, *generic patterns* are based on the presence of first person singular/plural personal pronoun. These patterns are found as strong indicator of self-referential tweets. We consider such self-referential tweets as explicit, otherwise implicit.

- Identification of clusters from explicit self-referential tweets: We identify explicit self-referential tweets clusters based on overlapping contents (i.e., tri-grams) and using Jaccard similarity between the explicit self-referential tweets.

- Pattern-mining from clusters: Once the explicit self-referential tweets clusters are identified, we fetch the most frequent substring (i.e, tri-gram) as a referential pattern from each cluster.

- Identification of implicit self-referential tweets: If an implicit tweet matches with the referential pattern of any cluster, then it is considered as a self-referential tweet.

- Merge with explicit tweets: Finally, all identified implicit self-referential tweets are merged with explicit tweets to generate a list of the self-referential tweets.

Once the list of self-referential tweets is generated, it is passed to the model learning and classification module for self-deprecating sarcasm detection. To this end, each self-referential tweet is converted into an input vector, it is fed to pre-trained `GloVe` word embedding, and model learning and classification task is accomplished using LSTM for detecting self-deprecating sarcasm.

This remainder of this paper is organized as follows. Section 2 presents a brief review of the state-of-the-art techniques and approaches for computational sarcasm detection. It also highlights the uniqueness of our proposed approach over the existing state-of-the-art techniques. Section 3 presents the functional details of the proposed approach, including model learning and classification using LSTM. Section 4 presents the experimental and evaluation results. Finally, section 5 concludes the paper and discusses future research directions.

2 Related Work

Automatic sarcasm detection is considered as a classification task (Zhang et al., 2016), and the main task is to classify any piece of texts as sarcasm or non-sarcasm. Tsur et al. (2010) applied semi-supervised approach to detect sarcasm in Amazon product reviews. Davidov et al. (2010) applied the same approach to detect sarcasm in tweets and product reviews. González-Ibánez et al. (2011) considered lexical and pragmatics features to detect sarcasm on Twitter datasets.

[4] `https://bit.ly/2EEuQGQ` (last accessed on 15-Nov-19)

Riloff et al. (2013) identified sarcastic contrast-based patterns and considered words with positive sentiment and negative phrases in a tweet containing sarcasm. Liebrecht et al. (2013) discussed the role of hyperrbole in sarcasm detection. Ptácek et al. (2014) detected sarcasm in English and Czech tweets. Bharti et al. (2015) proposed rule-based algorithms based on some patterns for sarcasm detection. They also highlighted the importance of hyperbole in sarcastic texts. Bamman and Smith (2015) extracted extra-linguistic information based on the context of the instances for sarcasm detection.

Rajadesingan et al. (2015) applied three machine learning classifiers – Support Vector Machine (SVM), logistic regression, and decision tree for sarcasm detection, considering the behavioral modeling-based approach. Ghosh et al. (2015) proposed SemEval-2015 (task-11) and considered sarcasm, irony, and metaphor for sentiment analysis in Twitter data. Joshi et al. (2015) discussed the role of incongruity for sarcasm detection. Bouazizi and Ohtsuki (2016) considered a pattern-based approach. Mishra et al. (2016) considered lexical- and contextual-based features. Joshi et al. (2016) proposed word-embedding related features using `Word2Vec`[5].

Recently, deep learning models have been used as a popular technique for sarcasm detection problem. Zhang et al. (2016) applied a bi-directional gated recurrent neural network for sarcasm detection. They considered syntactic and semantic information and extracted contextual features. Amir et al. (2016) applied content- and user embedding-based Convolutional Neural Network (CNN) model. Ghosh and Veale (2016) considered CNN, LSTM, and Deep Neural Network (DNN) for sarcasm detection. Poria et al. (2016) considered features, such as sentiment, emotion, and personality and applied SVM and CNN classifiers. Tay et al. (2018) considered attention-based neural model for sarcasm detection. Hazarika et al. (2018) proposed a contextual sarcasm detector using CNN-based textual model in which context and content related information are used for sarcasm detection. Recently, Dubey et al. (2019a) converted sarcastic texts into non-sarcastic interpretation using encoder-decoder, attention, and pointer generator architectures. Dubey et al. (2019b) detected sarcasm in numerical portion of tweets using CNN and attention network.

Though sarcasm detection is widely covered by the researchers, studies related to the varied categories of sarcasm are still not explored. Recently, Abulaish and Kamal (2018) noticed the use of self-deprecating sarcasm in Twitter, mainly for the purpose of brand endorsement and sales campaign. They considered self-deprecating sarcasm as a special category of sarcasm in which users express sarcasm over themselves. They also proposed a rule-based and machine learning-based approach for detecting self-deprecating sarcasm detection in Twitter. The proposed work in this paper is new LSTM-based deep learning approach for self-deprecating sarcasm detection in textual data.

3 Proposed Approach

In this section, we discuss the proposed LSTM-based deep learning approach for self-deprecating sarcasm detection. Figure 1 presents the workflow of the proposed approach. It can be seen from this figure that besides data crawling and data pre-processing, the main functionalities of the proposed approach are self-referential tweets detection, and self-deprecating sarcasm detection using deep learning technique. Further details about all functional modules are presented in the following sub-sections.

3.1 Data Crawling

The data crawling module aims to retrieve English tweets using Twitter's `REST` API and it is implemented in `Python 2.7`. We have considered tweet ids provided as a part of two benchmark datasets – Ptácek et al. (2014) and SemEval-2015[6] to curate tweets using our data crawling module. In addition, we have also created our own Twitter dataset containing tweets crawled for the period 1st April 2019 to 19th May 2019.

3.2 Data Pre-Processing

The data pre-processing module aims to apply various pre-processing tasks on the curated tweets to produce fine-grained data for self-deprecating sarcasm detection. The pre-processing consists of data cleaning (removal of dots, retweets, numbers, hashtags, emoticons, @mention, URL's, am-

[5] `https://code.google.com/archive/p/word2vec/` (last accessed on 15-Nov-19)

[6] `https://bit.ly/34OnGgB` (last accessed on 15-Nov-19)

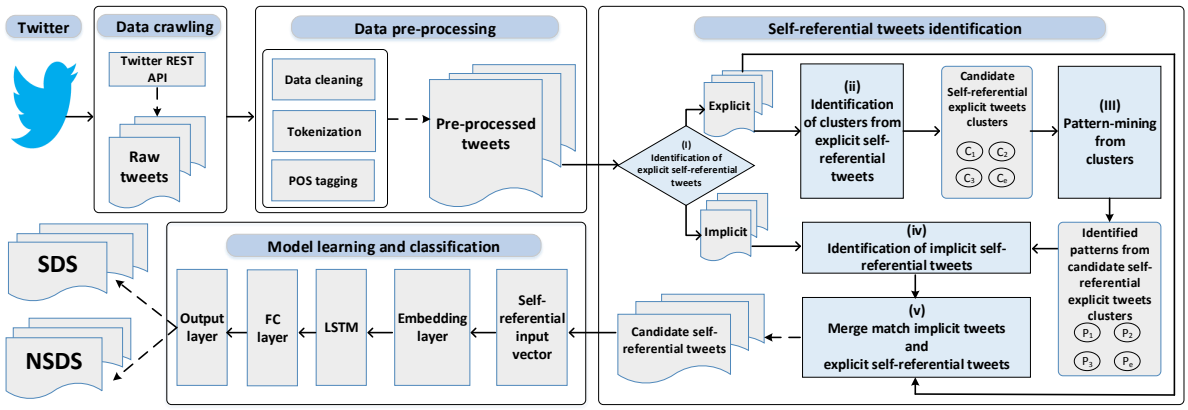


Figure 1: Work-flow of the proposed approach

persands, double quotes, and extra white spaces) and lower-case conversion. Thereafter, `spacy`[7] is used to tokenize the tweets and generate POS tags for each token.

3.3 Self-Referential Tweets Identification

After an in-depth analysis of the datasets it is observed that all tweets are not self-referential or self-deprecating in nature. To this end, this module presents a filtration mechanism to generate a corpus of self-referential tweets. The non-self-referential tweets are filtered from further consideration because they rarely contain a self-deprecating sarcasm. Motivated by Zhao et al. (2015), identification of self-referential tweets is performed using the following sequence of steps.

(i) ***Identification of Explicit Self-Referential Tweets:***
In this step, we identify the self-referential tweets that have explicit pattern in the text and these tweets are considered for further processing to mine implicit patterns (signals) of self-referential behavior in tweets. The explicit self-referential tweets have certain patterns, which can be defined using the regular expressions given in Table 1. The tweets from the pre-processed corpus are matched using these regular expressions to identify the explicit self-referential tweets. The pattern for explicit nature of self-referential tweets are of two types – *specific* and *generic*.

The *specific patterns* are based on either sequential order of tokens and tags, or sequential order of tokens. If any of the specific pattern

[7] `https://spacy.io/` (last accessed on 15-Nov-19)

Patterns	Category
UH $(i \mid my)$	Specific
$(we \mid i)$ [love] $(it \mid when)$	Specific
$when$ $(my \mid our)$	Specific
$(am \mid are)$ $[still]$	Specific
$(i \mid my \mid me \mid mine \mid myself)$	Generic
$(we \mid are \mid us \mid our \mid ourselves)$	Generic

Table 1: Regular expressions to identify explicit self-referential tweets

from table 1 founds in the pre-processed tweets, then it is added to the explicit set, otherwise it is checked further from *generic patterns*. The *generic patterns* are based on the first person singular/plural personal pronoun, such as 'i', 'we', and their objective and possessive cases, such as 'my', 'me', '$mine$', '$myself$', 'are', 'our', 'us', and '$ourselves$'. The first person singular/plural personal pronoun and its grammatical variants are strong indicator for a tweet to be referred as self-referential.

If any of the token from the pre-processed tweet matches with any *generic patterns*, then such tweet is considered as explicit self-referential tweet, and added to the explicit set of self-referential tweets, E_s. Otherwise, the tweet is added to the set of implicit tweets, I_t. Further, the identified explicit tweets are modeled as a undirected weighted graph and given to a clustering algorithm for further processing, which is defined in the next step.

(ii) ***Identification of Clusters from Explicit Self-Referential Tweets:***
This step clusters the tweets in E_s to identify the near-duplicate (similar) explicit self-referential tweets. To this end, first E_s tweets are modeled

as an undirected graph, where each node of the graph represents a tweet and edge represents the similarity between the underlying pair of nodes. The similarity between two tweets (nodes), say t_i and t_j, is calculated using Jaccard coefficient to observe the overlapping set of tri-grams between the tweets, as defined in equation 1, where T_i and T_j represents the set of tri-grams for tweets t_i and t_j, respectively. We choose tri-grams in our experiment because self-deprecating phrases in a tweet generally contain at least three words. We create an edge between a pair of near-duplicate tweets if the Jaccard similarity based on set of tri-grams is greater than a threshold 0.6 as defined in (Zhao et al., 2015). Thereafter, `depth first search` algorithm is applied on the constructed graph to extract clusters (connected components), where each cluster represents the set of identical explicit self-referential tweets. The extraction process only extract clusters having atleast three tweets.

$$J(t_i, t_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|} \quad (1)$$

(iii) ***Pattern-Mining from Clusters:***
Following the cluster identification process in the previous step, this step mines frequent patterns from the extracted clusters. To this end, the occurrence probability of every pattern of each cluster is computed and patterns having probability greater than 0.8 are regarded as patterns. For example, if a cluster has 5 tweets and a tri-gram "great way start" occurs in four out of 5 tweets, then it can be regarded as a frequent pattern (tri-gram). This procedure is repeated for every pattern in each cluster to extract the list of frequent patterns. Thereafter, the duplicate frequent patterns identified from two or more clusters are filtered to generate unique set of frequent patterns P.

(iv) ***Identification of Implicit Self-Referential Tweets:***
The first step of this whole procedure held tweets which have no explicit pattern as self-referential tweets, called *implicit tweets*. This step will improve the recall of the self-referential tweets identification process. This step matches the identified patterns from previous step in *implicit tweets* to extract implicit self-referential tweets. To this end, first an implicit tweet is tokenized in to tri-grams and thereafter these set of tri-grams are matched with the set of frequent patterns P using Jaccard similarity. Finally, a tweet that has Jaccard similarity greater than a threshold 0.6 is considered as a implicit self-referential tweets. This procedure is repeated for every tweets of I_t to generate a set of implicit self-referential tweets, I_s. For example, table 2 presents 3 example implicit self-referential tweets identified from I_t.

Pattern matched implicit self-referential tweets
1. great way start nothing.
2. waking with stomach pains best way start day.
3. battling cousin always great way end day.

Table 2: Implicit self-referential tweets identified from I_t

(v) ***Merging of Implicit and Explicit Tweets:***
Finally, in this step, the identified implicit self-referential tweets are added to the set of explicit self-referential tweets to generate a final set of self-referential tweets i.e. $S = E_s \cup I_s$. In the remaining paper, this curated corpus of self-referential tweets is used for experimental evaluation.

3.4 Model Learning and Classification

In recent years, deep learning has become an emerging trend in the field of text mining and natural language processing. The semantic modeling of textual data using deep learning approaches has drawn significant attention among the research community. Various neural network-based models including CNN, LSTM, and DNN are used for diverse text modeling applications such as document classification, machine translation, speech recognition, and so on. Detection of self-deprecating sarcasm is one such application that is largely unexplored. To this end, we modeled the self-deprecating sarcasm detection as a deep-learning problem.

On analysis, it is found that the long sequence of words or phrases plays an important role to construct a self-deprecating sarcastic patterns, such as *love being ignored*, *office on sunday*, and *happy to be late* in a tweet. Therefore, to model the long-sequences based self-deprecating sarcastic pattern, LSTM seems a perfect fit. Using model learning through LSTM, a self-referential tweet is classified as a Self-Deprecating Sarcasm (SDS) or Non Self-Deprecating Sarcasm (NSDS).

A detailed discussion about the model learning and classification is presented in following subsection.

Input Layer:
In this layer, a self-referential tweet, containing n words, is given as an input. In this manner, each self-referential tweet is converted to a self-referential input vector where every word is replaced with its index value of the dictionary, i.e., $S \epsilon R^{1 \times n}$. Further, each self-referential input vector is padded and converted in the matrix form. The padding is used to make every input of same length. Thereafter, padded input vector is passed to the next layer (i.e., embedding layer).

Embedding Layer:
In the padded vector from the input layer, all the words are replaced with their corresponding representation vector or embeddings. In this paper, we have used pre-trained `GloVe` 200-dimensional embeddings trained on a Twitter corpus of 27 billion tokens. As a result of this procedure, the self-referential input tweet matrix is converted to $S \epsilon R^{L \times D}$, where L is the maximum tweets length and D represents embedding dimension.Thereafter, the embedding layer output is passed to the LSTM layer.

LSTM:
Hochreiter and Schmidhuber (1997) proposed LSTM architecture, which is a type of RNN. It is easier to train an LSTM model in comparison to an RNN model. Moreover, it also overcomes the vanishing gradient problem while back propagation through time. In LSTM, the long term temporal dependencies can be easily captured between two time steps using the memory cell. Figure 2 presents the architecture of LSTM, where each memory cell consists of *input gate* i_t, *forget gate* f_t, and *output gate* o_t. These digital gates are responsible for memory update mechanism, and it acts as a function for the current input x_t and previous hidden state h_{t-1}.

An LSTM model is trained using equations 2, 3, 4, 5, 6, and 7. Equations 2 and 3 present *input* and *forget* gates, whereas equations 5, 6, and 7 present *output gate*, *new cell state*, and *hidden state*, respectively.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2}$$

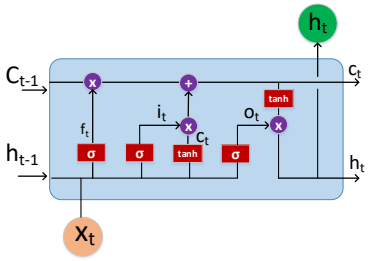


Figure 2: The architecture of LSTM

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3}$$

$$q_t = \tanh(W_q[h_{t-1}, x_t] + b_q) \tag{4}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_0) \tag{5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot q_t \tag{6}$$

$$h_t = o_t \odot \tanh(c_t) \tag{7}$$

In equation 4, the non-linear activation function – tanh is used to squash the value between -1 and 1, and it plays a role for cell state to forget the memory. On the other hand, non-linear activation function, sigmoid (σ) generates an output in the interval [0, 1]. LSTM works as the *gating* function for the three gates, which are discussed in the previous paragraph. Since it has a value in interval [0, 1], the information across the gates are either passed completely or not.

FC and Output Layers:
The output from the LSTM layer is passed to the fully connected dense layer followed by a sigmoid activation function. We have used binary cross-entropy as the loss function, used 40 epochs for training the model, batch-size of 256, verbose is 2, and `adam` as an optimizer. The dataset is divided into training and testing parts for experimental evaluations wherein 80% of the data is used for training and remaining 20% is used for testing procedure.

4 Experiments Setup and Results

In this section, we discuss the experimental evaluation of the proposed approach.

4.1 Experimental Settings

We have implemented the experimental setup for data crawling, data pre-processing, and self-referential tweets identification tasks in `Python 2.7`, model training and classification tasks in `Python 3.5`, and used `Keras` neural network API for LSTM model. Table 3 presents the hyper-parameters values of LSTM model used in the proposed approach.

Hyper-parameters	Value
Embedding dimension	200
Padding sequences	20
Spatial dropout (after embedding layer)	0.4
Number of neurons	256
Dropout (after LSTM layer)	0.4

Table 3: Hyper-parameters values for LSTM model used in our proposed approach

4.2 Datasets

The proposed approach is evaluated over three Twitter datasets including two benchmark datasets by Ptácek et al. (2014) and SemEval-2015. The authors released only tweet-ids for these benchmark datasets due to privacy concerns. Therefore, a crawler is developed in `Python 2.7` to curate tweets corresponding to provided tweet-ids using Twitter `REST` API. However, few tweets were deleted or protected and, as a result, we were unable to crawl all the tweets. A brief statistics about these two datasets is given in the first two rows of table 4. Apart from the two benchmark datasets, we curated a Twitter dataset from 1st April to 19th May 2019 using "#sarcasm" hashtag. We refer this dataset as `Twitter-280` and its statistical summary is given in the third row of table 4. Similarly, we crawled non-sarcastic tweets using two #not, #hate hastags. Table 5 presents the statistics of identified self-referential tweets after the *self-referential tweets identification* module. Table 6 presents the final statistics of the balanced and unbalanced datasets generated from table 5.

Datasets	#Sarcasm	#Non-sarcasm	Total (#tweets)
Ptácek et al. (2014)	53088	98195	151283
SemEval-2015	1526	2366	3892
Twitter-280	13786	14949	28735
Total (#tweets)	68400	115510	183910

Table 4: Statistics of the crawled datasets

Datasets	#Sarcasm	#Non-sarcasm	Total (#tweets)
Ptácek et al. (2014)	29580	37767	67347
SemEval-2015	761	1609	2370
Twitter-280	6971	7017	13988
Total (#tweets)	37312	46393	83705

Table 5: Statistics of identified self-referential tweets by the *self-referential tweets identification* module

4.3 Evaluation Metrics

This section discusses the standard data mining metrics – *precision*, *recall*, and *f-score*, which are used to evaluate the proposed approach. Formally, these metrics in terms of True Positives (*TP*), False Positives (*FP*), and False Negatives (*FN*) are define in equations 8, 9, and 10, where *TP* is defined as the number of correctly classified as SDS tweets, *FP* is defined as number of NSDS tweets misclassified as SDS tweets, and *FN* is defined as number of SDS tweets misclassified as NSDS tweets.

$$Precision\ (\pi) = \frac{TP}{FP + TP} \tag{8}$$

$$Recall\ (\rho) = \frac{TP}{FN + TP} \tag{9}$$

$$F\text{-}score\ (F1) = \frac{2 \times \pi \times \rho}{\pi + \rho} \tag{10}$$

4.4 Evaluation Results

This section presents the experimental evaluation results over the three datasets discussed in subsection 4.2. All the experimental evaluations are performed using an LSTM model trained on 40 epoch. Table 7 presents the performance evaluation results of our proposed approach using the LSTM model on balanced and unbalanced datasets in terms of three evaluation metrics. On analysis, it can be observed from this table that in terms of all the three evaluation metrics, the proposed approach performs comparatively better on balanced datasets and shows slightly lower performance on unbalanced datasets. Another interesting observation from this table is that, in terms of all the three evaluation metrics, proposed approach performs best on Ptácek et al. (2014) dataset. Further, table 7 shows that the proposed approach performs comparatively better on the balanced version of our created dataset.

Datasets		#Sarcasm	#Non-sarcasm	Total (#tweets)
Ptácek et al. (2014)	Balanced	14500	14500	29000
	Unbalanced	5750	23000	28750
SemEval-2015	Balanced	500	500	1000
	Unbalanced	250	1100	1350
Twitter-280	Balanced	5000	5000	10000
	Unbalanced	500	2000	2500

Table 6: Statistics of the balanced and unbalanced datasets generated from table 5

Datasets		Evaluation results		
		π	ρ	F1
Ptácek et al. (2014)	Balanced	0.93	0.94	0.93
	Unbalanced	0.92	0.89	0.90
SemEval-2015	Balanced	0.86	0.84	0.85
	Unbalanced	0.93	0.75	0.83
Twitter-280	Balanced	0.90	0.92	0.93
	Unbalanced	0.89	0.86	0.88

Table 7: Performance evaluation of our proposed approach using LSTM on balanced and unbalanced datasets presented in table 6

4.5 Comparative Analysis

To be the best of authors knowledge there is no prior work on self-deprecating sarcasm detection using deep learning approach. However, a rule and machine learning-based approach was presented by the authors in Abulaish and Kamal (2018) and proposed approach is compared with that one. In Abulaish and Kamal (2018), authors considered Ptácek et al. (2014) dataset to detect self-deprecating sarcasm in tweets. We implemented Abulaish and Kamal (2018) to evaluated its efficacy over the three datasets. Figures 3 and 4 present the comparative performance evaluation of the proposed approach with Abulaish and Kamal (2018) in terms of *precision*, *recall*, and *f-score* over balanced and unbalanced version of all the three datasets, respectively.

It can be observed from figures 3 and 4 that the proposed LSTM-based deep learning approach outperforms Abulaish and Kamal (2018) in terms of *precision*, *recall*, and *f-score* on both balanced and unbalanced datasets. However, Abulaish and Kamal (2018) reported slightly better performance in terms of *precision* and *f-score* results on Ptácek et al. (2014) dataset.

5 Conclusion and Future Work

In this paper, we have proposed a new approach using LSTM-based deep learning for detecting self-deprecating sarcasm in textual data. The self-deprecating sarcasm is a special category of sarcasm in which users apply sarcasm on themselves. One of the major applications of this work is to promote self-deprecating marketing strategies. The proposed approach is evaluated over three Twitter datasets, including two benchmark datasets, and the experimental results are promising. It also performs significantly better than one of the state-of-the-art methods, which used rule-based and machine learning techniques for self-deprecating sarcasm detection. Exploring new patterns and consideration of multimedia contents for self-deprecating sarcasm detection seems one of the promising directions of future research.

Acknowledgments

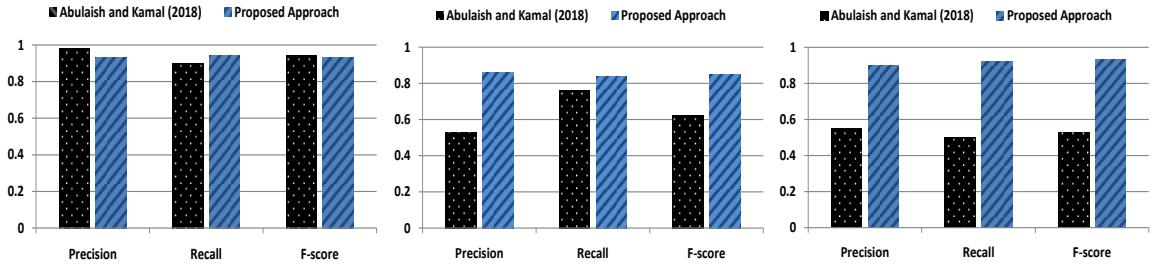
This publication is an outcome of the R&D work undertaken project under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, Government of India, being implemented by Digital India Corporation.

References

Muhammad Abulaish and Ashraf Kamal. 2018. Self-deprecating sarcasm detection: An amalgamation of rule-based and machine learning approach. In *Proceedings of the International Conference on Web Intelligence (IEEE/WIC/ACM), Santiago, Chile*, pages 574–579. IEEE.

Silvio Amir, Byron C. Wallace, Hao Lyu, Paula Carvalho, and Mário J. Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. In *Proceedings of the 20th Special Interest Group on Natural Language Learning Conference on Computational Natural Language Learning((SIGNLL-CoNLL), Berlin, Germany*, pages 167–177. Association for Computational Linguistics.

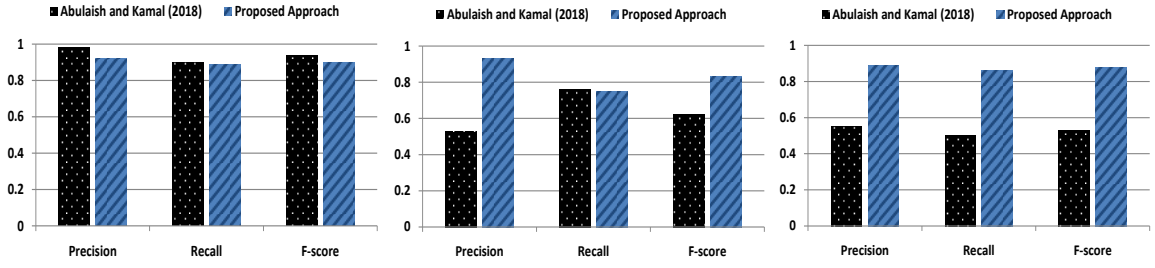
David Bamman and Noah A. Smith. 2015. Contextualized sarcasm detection on twitter. In *Proceedings of the 9th International Association for the Advancement of Artificial Intelligence Conference on Web and Social Media (ICWSM), Oxford, UK*, pages 574–577. Citeseer.



(a) (b) (c)

Figure 3: Visualization of the performance comparison results presented in table 7 over the balanced datasets (a) Ptácek et al. (2014) (b) SemEval-2015 (c) Twitter-280

.



(a) (b) (c)

Figure 4: Visualization of the performance comparison results presented in table 7 over the unbalanced datasets (a) Ptácek et al. (2014) (b) SemEval-2015 (c) Twitter-280

.

Santosh Kumar Bharti, Korra Sathya Babu, and Sanjay Kumar Jena Jena. 2015. Parsing-based sarcasm sentiment recognition in twitter data. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Paris, France*, pages 1373–1380. IEEE.

Mondher Bouazizi and Tomoaki Ohtsuki. 2015. Opinion mining in twitter how to make use of sarcasm to enhance sentiment analysis. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Paris, France*, pages 1594–1597. IEEE.

Mondher Bouazizi and Tomoaki Ohtsuki. 2016. A pattern-based approach for sarcasm detection on twitter. *IEEE Access*, 4:5477–5488.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the 14th Conference on Computational Natural Language Learning (CoNLL), Uppsala, Sweden*, pages 107–116. Association for Computational Linguistics.

Abhijeet Dubey, Aditya Joshi, and Pushpak Bhattacharyya. 2019a. Deep models for converting sarcastic utterances into their non sarcastic interpretation. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data (CoDS-COMAD), Kolkata India*, pages 289–292. ACM.

Abhijeet Dubey, Lakshya Kumar, Arpan Somani, Aditya Joshi, and Pushpak Bhattacharyya. 2019b. "when numbers matter!": Detecting sarcasm in numerical portions of text. In *Proceedings of the 10th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Medi Analysis (NAACL), Minneapolis, USA*, pages 72–80. Association for Computational Linguistics.

Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, John Barnden, and Antonio Reyes. 2015. Semeval-2015 task 11: Sentiment analysis of figurative language in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval), Denver, Colorado*, pages 470–478. Association for Computational Linguistics.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of the 15th North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), San Diego, California, USA*, pages 161–169. Association for Computational Linguistics.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twit-

ter: A closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL), Portland, Oregon*, pages 581–586. Association for Computational Linguistics.

Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea. 2018. Cascade: Contextual sarcasm detection in online discussion forums. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING), Santa Fe, New Mexico, USA*, pages 1837–1848. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd nnual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP), Beijing, China*, page 757–762. Association for Computational Linguistics.

Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. 2016. Are word embedding-based features useful for sarcasm detection? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Austin, Texas*, pages 1006–1011. Association for Computational Linguistics.

Ashraf Kamal and Muhammad Abulaish. 2019. Self-deprecating humor detection: A machine learning approach. In *Proceedings of the 16th International Conference of the Pacific Association for Computational Linguistics (PACLING), Hanoi, Vietnam*, pages 1–13. Springer.

Christine Liebrecht, Florian Kunneman, and Antal V. D. Bosch. 2013. The perfect solution for detecting sarcasm in tweets #not. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA), Atlanta, Georgia*, pages 29–37. ACL.

Abhijit Mishra, Diptesh Kanojia, and Pushpak Bhattacharyya. 2016. Predicting readers' sarcasm understandability by modeling gaze behavior. In *Proceedings of the 13th Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence, Phoenix, Arizona, USA*. AAAI.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. 2016. A deeper look into sarcastic tweets using deep convolutional neural networks. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING) Osaka, Japan*, pages 1601–1612.

Tomás Ptácek, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on czech and english twitter. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING), Dublin, Ireland*, pages 213–223.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the 8th Association for Computing Machinery International Conference on Web Search and Data Mining (WSDM), Shanghai, China*, pages 97–106. ACM.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra D. Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Seattle, Washington, USA*, pages 704–714. Association for Computational Linguistics.

Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. 2018. Reasoning with sarcasm by reading in-between. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), Melbourne, Australia*, pages 1010–1020. Association for Computational Linguistics.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *Proceedings of the 4th International Association for the Advancement of Artificial Intelligence Conference on Weblogs and Social Media (ICWSM), Washington, DC, USA*, pages 162–169. AAAI.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Tweet sarcasm detection using deep neural network. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING), Osaka, Japan*, pages 2449–2460.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web (WWW), Florence, Italy*, pages 1395–1405.

Unsung Challenges of Building and Deploying Language Technologies for Low Resource Language Communities

Pratik Joshi[1] **Christain Barnes**[2*] **Sebastin Santy**[1] **Simran Khanuja**[1]
Sanket Shah[1] **Anirudh Srinivasan**[1] **Satwik Bhattamishra**[1]
Sunayana Sitaram[1] **Monojit Choudhury**[1] **Kalika Bali**[1†]
[1] Microsoft Research, Bangalore, India
[2] Stanford University

Abstract

In this paper, we examine and analyze the challenges associated with developing and introducing language technologies to low-resource language communities. While doing so, we bring to light the successes and failures of past work in this area, challenges being faced in doing so, and what they have achieved. Throughout this paper, we take a problem-facing approach and describe essential factors which the success of such technologies hinges upon. We present the various aspects in a manner which clarify and lay out the different tasks involved, which can aid organizations looking to make an impact in this area. We take the example of Gondi, an extremely-low resource Indian language, to reinforce and complement our discussion.

1 Introduction

Technology pervades all aspects of society and continues to change the way people access and share information, learn and educate, as well as provide and access services. Language is the main medium through which such transformational technology can be integrated into the socioeconomic processes of a community. Natural Language Processing (NLP) and Speech systems, therefore, break down barriers and enable users and whole communities with easy access to information and services. However, the current trend in building language technology is designed to work on languages with very high resources in terms of data and infrastructure.

Also, as Machine Learning (ML) and NLP practitioners, we get caught up in an information-theoretic view of the problem, e.g., focusing on incremental improvements of performance on benchmarks or capturing accurate distributions over data, and tend to forget that the raison d'łtre of NLP is to build systems that add value to its users (Ruder, 2019). We want to build models that enable people to read the news that was not written in their language, ask questions about their health when they do not have access to a doctor, etc. And while these technology applications are more and more ubiquitous for languages with a lot of data, a larger majority of languages remain resource-poor and bereft of such systems. As discussed in the United Nations e-government survey (Nations, 2014), one of the most important obstacles to e-inclusion, particularly among vulnerable groups with little education, is language. Thus, by excluding these languages from reaping the benefits of the advancements in language technology, we marginalize the already vulnerable groups even further.

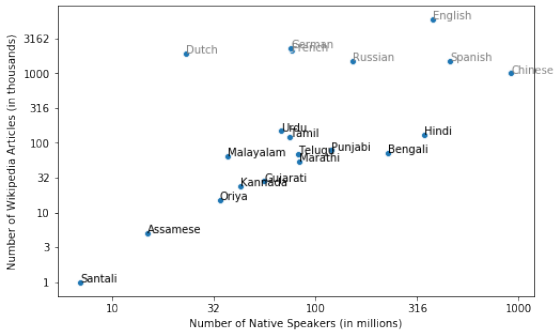


Figure 1: The points represent the disparity between number of wikipedia articles in comparison with the number of native speakers for a particular language.

India is a highly multilingual society and home to some of the largest language communities in the world. 6 out of 20 most-spoken (native) languages in the world are Indic. Ethnologue (Simons and Fennig, 2017) records 461 tongues in India out of 6912 worldwide (6%), the 4th largest belonging to any single country in the world. 122 of these

*Work done during internship at Microsoft Research
†Email: kalikab@microsoft.com

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 211–219
Hyderabad, India, December 2019.

languages are spoken by more than 10,000 people. 29 languages have more than 1 million speakers, which include indigenous tribal languages like Gondi and Mundari, some without a supported writing system or script. Despite the large numbers of users, most of these languages have very little data available. Figure 1 shows that as compared to some of the much lesser spoken languages like German, Indic languages are severely low resourced. In a vast country like India, access to information thus becomes a huge concern. This lack of information means that not only do these communities not have information in domains like agriculture, health, weather etc., which could improve their quality of lives, but they may also not be aware of their basic rights as citizens of the country.

In this paper, we take the position that the current direction of advanced language technology towards extremely high data requirements can have severe socio-economic implications for a majority of language communities in the world. We focus on specific aspects of designing and building systems and applications for low resource languages and their speech communities to exemplify viable social impact through language technology. We begin by discussing the aspect of information exchange, which is the core motivation behind enabling low-resource language communities. We then steer our analysis towards the design and creation of an interface for people in these communities to simplify and enrich the process of information exchange. Finally, we gather insights about how to deploy these technologies to ensure extensive impact by studying and taking inspiration from existing technological deployments.

We use Gondi, a South-Central Dravidian language in the vulnerable category on UNESCO's Atlas of the Worlds Languages in Danger (Moseley, 2010), as an example wherever possible. Spoken by nearly 3 million people (India, 2011) in the Indian states of Chhattisgarh, Andhra, Odisha, Maharashtra and Karnataka, it is heavily influenced by the dominant state language. However, it is also one of the least resourced languages in India, with very little available data and technology.

We believe that the components discussed in the sections below encapsulate the spectrum of issues surrounding this field and that all future discussions in this area will also fall under the umbrella of these categories. We believe that by focusing on Gondi, we will not only empower the Gondi community but more importantly, understand and create a pipeline or framework which can serve as a clear guide for potential ventures which plan on introducing disruptive language technologies in under-served communities.

2 Information Exchange

The primary element in communication is information exchange. People living in less connected areas are often unable to get the kind of information they need, due to various socio-economical and technological barriers. As a result, they miss out on crucial knowledge required to improve their well-being. There are three co-dependent aspects woven into the fabric of information exchange - access of information, quality and coverage of the information and methods to create and digitize available knowledge (generation).

2.1 Access

This section refers to past work and current ventures of making digital resources adequately available and accessible to people.

2.1.1 Making Information Accessible to People

Less-connected and technologically underdeveloped areas often suffer from the limited accessibility of up-to-date information. Providing more individuals access to the online repositories of information can often help them improve their well-being.

There are some situations particularly during natural calamities where the absence of notifications about potentially disaster-prone areas can result in life and death situations of individuals. People in regions with sparse connectivity often fall victim to these incidents due to lack of timely updates. Using technical platforms to support the spread of information to these regions is an important goal to keep in mind. LORELEI (Strassel and Tracey, 2016) is a DARPA funded initiative with the goal of the building of technologies for dealing and responding to disasters in low resource language communities. Similar initiatives in India would be capable of saving lives.

The daily function and health of individuals in a community can be influenced positively by the dissemination of relevant information. For example, healthcare and agricultural knowledge

can affect the prosperity of a rural household, making them aware of potential solutions and remedies which can be acquired. There has been a considerable body of work focused on technology for healthcare access, which includes telemedicine (Brauchli et al., 2005) and remote diagnosis (Surana et al., 2008). While the use of telecenters to spread information on agricultural practises has been employed, persuading users to regularly use the telecenters (Ramamritham et al., 2006) is a challenge, which could be addressed by the use of language technologies to simplify access. VideoKheti (Cuendet et al., 2013) is an example of a voice-based application which provides educational videos to farmers about effective agricultural practices. Similar studies have been carried out to assess the effectiveness of voice-activated applications for farming (Patel et al., 2010). There are considerable challenges, however, to ensure that these solutions are inclusive and accessible to low-literate and less-connected users.

Similarly, there are situations where there are certain rights and duties which an individual as a citizen of India is entitled to. Some communities have long been exploited and ill-treated (Ganguly and Chaudhary, 2003), and providing them information regarding their rights as well as accurate news could foster a sense of solidarity within the community and encourage them to make their voice heard. An extensive study on the impact of CGNet Swara (Marathe et al., 2015) showed that this citizen journalism platform inspired people in rural communities, gave them a feeling of being heard, and provided a venue to voice their grievances. There are also other promising ventures such as Awaaz De (Patel et al., 2010) and Gram Vaani (Moitra et al., 2016) which aim to boost social activism in a similar manner.

2.1.2 Making more digital content available

The process of enabling more low-resource language communities with tools to access online information alone is not sufficient. There need to be steps taken to make more of the content which exists online interpretable to people in these communities. For example, The Indian Constitution and other similar official communications from the government are written in 22 scheduled languages of India. Lack of access to other related documents deprives them of basic information. This is where building robust machine translation tools for low resource languages can help. Cross-language information retrieval makes extensive use of these translation mechanisms (Zhou et al., 2012) where information is retrieved in a language different from the language of the user's query. McNamee and Mayfield (2002) describes a system making use of minimal resources to perform the same.

There is huge potential for language technologies to be involved in content creation and information access. Further, more accurate retrieval methods can help the user get relevant information specific to their needs and context in their own language.

2.1.3 Making NLP models more accessible to low resource languages

Often, many state-of-the-art tools cannot be applied to low-resource languages due to the lack of data. Table 1 describes the various technologies and their presence concerning languages with different levels of resource availability and the ease of data collection. We can observe that for low resource languages, there is considerable difficulty in adopting these tools. Machine Translation can potentially be used as a fix to bridge the gap. Translation engines can help in translating documents from minority languages to majority languages. This allows the pool of data to be used in a number of NLP tasks like sentiment analysis and summarization. Doing so allows us to leverage the existing body of work in NLP done on resource-rich languages and subsequently apply it to the resource-poor languages, thereby foregoing any attempt to reinvent the wheel for these languages. This ensures a quicker and wider impact.Wan (2008) performs sentiment analysis on Chinese customer reviews by translating them to English. They observe that the quality of machine translation systems are sufficient for sentiment analysis to be performed on the automatically translated texts without a substantial trade-off in accuracy.

2.2 Generation

This section refers to the generation of digital content which enriches online repositories with more diverse sets of information.

2.2.1 Digitization of Documents

There is a need to generate digital information and content for low-resource languages. It not only

Technology	Availability of technology for the resource status of a language				Data/Expertise Requirement		
	High	Moderate	Low	No	Linguistic Expertise	Unlabeled Data	Labeled Data
Input/Output Support							
Font & Keyboard	⋆⋆⋆	⋆⋆⋆	⋆⋆⋆	⋆⋆	⋆⋆⋆		
Speech-to-Text	⋆⋆⋆	⋆⋆			⋆	⋆⋆	⋆⋆⋆
Text-to-Speech	⋆⋆⋆	⋆⋆	⋆		⋆⋆⋆		⋆⋆
Text Prediction	⋆⋆⋆	⋆⋆⋆	⋆⋆			⋆⋆⋆	
Spell Checker	⋆⋆⋆	⋆⋆⋆	⋆⋆		⋆⋆⋆	⋆⋆	
Grammar Checker	⋆⋆⋆	⋆⋆			⋆⋆	⋆⋆⋆	⋆⋆
Local Language UI							
	⋆⋆⋆	⋆⋆⋆	⋆⋆		⋆⋆⋆		
Information Access							
Text Search	⋆⋆⋆	⋆⋆	⋆		⋆	⋆⋆⋆	⋆⋆
Machine Translation	⋆⋆	⋆	⋆		⋆⋆		⋆⋆⋆
Voice to Text Search	⋆⋆⋆	⋆				⋆	⋆⋆⋆
Voice to Speech Search	⋆⋆	⋆			⋆	⋆⋆⋆	⋆⋆⋆
Conversational Systems							
	⋆⋆	⋆			⋆⋆⋆	⋆⋆⋆	⋆⋆⋆

Table 1: Enabling language technologies, their availability and quality (⋆⋆⋆ - excellent quality technology, ⋆⋆ - moderately good but usable, ⋆ - rudimentary and not practically useful) for differently resourced languages, and their data/knowledge requirements (⋆⋆⋆ - very high data/expertise, ⋆⋆ - moderate, ⋆ - nominal and easily procurable). This information is based on authors' analysis and personal experience.

benefits the community by creating digital content for their needs, but it also provides data which can be used to train data-driven language technologies, such as ASRs, translation systems, and optical character recognition systems. Efforts to digitize content in India have been conducted in the past few years. The Government of India launched the Digital India initiative [1] in 2015, which aims to digitize government documents in one of India's 120+ local languages. Such initiatives have evidently been useful before. For instance, the IMPACT project [2] by the European Union was a large scale digitization project which helped push a lot of innovative work towards OCR and language technology for historical text retrieval and processing. IMPRINT is a similar initiative created by the Ministry of Human Resource Development (MHRD) to drive further research towards addressing such challenges.

The recent advancements in OCR technologies can propel efforts to digitize more handwritten documents. Such initiatives are already being undertaken to digitize and revive historical languages in Japan (Clanuwat et al., 2018). Digital India library is a project that aims towards digitizing books and making them available online. Apart from printed books, a lot of ancient literature is written on palm leaves. The Regional Mega Scan Centre (RMSC) at IIIT Hyderabad has digitized over 100,000 books, one-third of which are in Indian Languages and additionally, they have also digitized text from scans of palm leaves. More initiatives such as these will help preserve and revive a number of languages that are part of the Indian heritage.

2.2.2 Crowdsourcing

Data collection via crowdsourcing can be a challenge for low resource languages, primarily due to the expensive nature of the task coupled with the lack of commercial demand for such data. Thus, collecting this data at low cost becomes an important priority. Project Karya is a crowdsourcing platform which provides digital work to low-income workers. Although the data quality can be a concern, promising results have shown otherwise. Chopra et al. (2019) tested the quality of crowdsourced data in rural regions of In-

[1] http://www.digitalindia.gov.in/
[2] http://www.impact-project.eu/

dia, tasking individuals with the digitization of Hindi/Marathi handwritten documents. A 96.7% accuracy of annotation was yielded, proving that there is potential in this area. Recently, collection of Marathi speech data is also being conducted. In a similar fashion, Navana Tech[3], a startup, has been collecting data in mid and low-resource languages of verbal banking queries so that they can be integrated into various banking application platforms for financial inclusion. Such crowdsourcing platforms not only act as a potential data for low-resource communities, they also benefit low-income workers by increasing their current daily wage. Such ventures would enhance the inclusion of such workers in the digitization process, something which aligns with the aims of the Digital India mission.

The collection of data in an extremely low-resource language like Gondi can be particularly tricky, additionally considering the fact that Gondi does not have an official script. Pratham Books [4] is a non-profit organization which aims to democratize access to books for children. They recently hosted a workshop where they trained members of the local community to translate books on StoryWeaver[5], their open-source publication platform. At the end of this workshop, approximately 200 books were translated from Hindi to Gondi (Devanagiri script). This was the first time children's books were made available in Gondi, and it also sparked the creation of parallel data for Hindi-Gondi translation systems.

3 Interface

The design of a user-friendly interface plays a very crucial role in ensuring that the deployed technology encompasses all strata of society. It is often seen that a majority of target users have not had the privilege of education, and show varying levels of literacy, both foundational and digital. In such scenarios, text-based modalities pose several limitations from both the user and designer perspectives, and graphical user interfaces have been the preferred choice in these applications. Thies et al. (2015) reports that text-based interfaces were completely redundant for illiterate users and severely error-prone for literate but novice users. Further, several languages do not have unique keyboard standards or fonts, and some do not have a script at all (Boyera, 2007).

To overcome these issues with text, speech as a modality has also been deployed with varying success. 'CGNet Swara', a citizen-run journalism portal, uses a phone-based IVR system to educate illiterate users (Mudliar et al., 2013). Avaaj Utalo allows users to make simple phone calls to ask questions or browse questions and answers asked on agricultural topics (Patel et al., 2010). Spoken Web is another application wherein users can create voice sites analogous to websites which can then be easily accessed through voice interaction on mobile phones (Kumar et al., 2010). These serve to provide farmers with relevant crop and market information. An attempt to leverage the complementarity of voice and graphic-based inputs was made by VideoKheti, a mobile system with a multi-modal interface for low-literate farmers providing agricultural extension videos on command in their own language or dialect (Cuendet et al., 2013). They report that people in these communities find it difficult to use softkey type keyboards that are extremely common on modern smartphones. Instead, they proposed a system comprising of large buttons, graphics and some voice input. Such a system for delivering information to farmers was made and they showed that the farmers were very comfortable using it. Their results also show that a speech interface alone was not enough for that scenario, except in cases where the search list was long and the results were dependent on keywords or short phrases. Similarly, the Adivasi Radio App [6], based on text-to-speech (TTS) technology, is developed to read out written reports in Gondi, one of the main tribal languages in Chhattisgarh. Bolo is another mobile application which uses a very simple interface to improve children's literacy in India. Project Karya also proposes to divide massive digital tasks into microwork and crowdsource this work to millions of people in rural India via phones [7].

While voice might solve the foundational literacy problems, the lack of digital literacy is often more challenging to overcome. Mondal (2019) demonstrate the use of an app to teach the Mundari language to children. The app comprised of a series of games designed with the help of the community. The content was delivered in the Bangla

[3] https://navanatech.in/
[4] https://prathambooks.org/
[5] https://storyweaver.org.in/
[6] AdivasiRadio - Google Play
[7] Project Karya

script, which was what the children were taught in school. Their study noted that children from such communities found the usage of a smartphone to be difficult.

Relying on voice-based systems also poses a few challenges. It is not easy to build robust ASR systems for these languages due to severe lack of data, dialect variations and several such constraints. An attempt to resolve this was made with the development of the SALAAM ASR (Qiao et al., 2010) which uses the acoustic model of an existing ASR and performs a cross-lingual phoneme mapping between the source and target language. This, however, is limited to recognition of a very small set of vocabulary, but finds use due to its' cost-effective and low resource setting.

4 Deployment and Impact

After developing technologies to provide information, and ensuring that the applications are designed in such a way that they are accessible to the population, the technology must be effectively deployed. Specialized applications are useless if they are not deployed properly in a way that accesses their end-users. When deploying a specially developed technology, the application must be deployed with consideration of the existing community dynamics. For any deployment to be successful, it usually must be able to be purposefully integrated into the lifestyle of community members - or have strong utilization incentives if it is a transformative technology. In this section, we will review examples of technology dissemination to low-resource/rural communities, and the impacts of effective deployment. While some technologies that we examine are not deployed utilizing low-resource languages specifically, the types of rural communities and villages in which they are deployed are analogous to the contexts in which low-resource languages exist, and clear parallels can be drawn.

Integrating the usage of a language technology intervention into a community in a low-resource context requires much more simply introducing the technology. Unlike hardware interventions and innovations like solar panels or new agricultural tools, language technologies often rely on the delivery, exchange, and utilization of information, which is much less tangible than physical solutions. This is especially for people with limited previous exposure to digital technology. Upon observing a selection of language-based interventions that were deployed in low-resource contexts, we observed that the most successful deployments of technologies tended to have three components of success. They: 1) Initially launched by seeding with target communities, 2) Worked closely to engage the community itself with the technology and information, and 3) Provided a strong incentive structure to adapt the technology - this incentive could be as simple as payments or as complex as communicated benefits from the technology.

4.1 Case Studies

In this section, we will be reviewing and comparing three separate technological systems, Learn2Earn (Swaminathan et al.), Mobile Vaani (Moitra et al., 2016), and the Climate and Agriculture Information Service (CAIS)(Christensen et al., 2019), and see how they utilized the rules of successful deployment outlined above. Learn2Earn, developed by Microsoft Research, is a simple IVR based mobile language technology app which uses quizzes to educate people and spread public awareness campaigns, launched initially in rural central India. Mobile Vaani is a large-scale and broad community-based IVR media exchange platform, developed by the NGO Gram Vaani. It currently has over 100,000 unique monthly active users, and processes 10,000 calls per day across the three Indian states of Bihar, Jharkhand, and Madhya Pradesh. Finally, the CAIS system is an SMS-based information delivery system designed for farmers who live in a rural, low-resource and no connectivity agricultural village on the Char Islands in the Bangladeshi Chalan Beel Wetland. This application provides weather data and agricultural advice to farmers on a periodic basis and was developed by a collaboration between mPower and two local NGOs. After designing the platform in accessible ways, each deployment process began with the seeding within a target community within itself. All examples that we studied became successful only after a small scale launch of their product. These launches occurred in different ways but were all based on targeting a starting group of users and incentivizing them to utilize and share the product. The initial users were people who were somewhat fluent in the technology (either through training or existing knowledge), and who knew of or had specialized needs that the technology could address.

4.1.1 Learn2Earn

Learn2Earn was built as a tech-enabled information dissemination system; its original information awareness campaign centred on informing farmers about their rights as guaranteed in Indias Forest Rights act. Because of the nature of their message content and delivery, the researchers decided to seed the platform with a single advertisement on an existing IVR channel already utilized by farmers. This advertisement reached 150 people, and provided them with distinct financial incentives to both call the platform, and invite friends to the platform. While only 17 of the original listeners of the advertisement went on to call the number, those respondents were members of the relevant community (farmers who were familiar with IVR technology) and were networked through family and friendships to additional ideal users of the platform. Within 7 weeks, the incentive structure allowed the platform to spread from the original 17 users to over 17,000, with little influence from the platform respondents. (Swaminathan et al.)

4.1.2 Mobile Vaani

Mobile Vaani initially tried to launch in 2011 by encouraging employees from their partner NGOs to distribute their platform. The platform was initially imagined as a voice-based, inclusive medium for communities to express their grievances and communicate with each other digitally. The initial employees who recruited for the platform were not from the community but did work closely with them regularly. While there was some initial success in the launch, the mobile Vaani Team were unable to grow at a significant pace because they informed the end-users about their intended design and usage of the technology, which set unrealistic expectations of the platform in the minds of the participating users after the technology could not be used in the exact way that it was encouraged. A few months later, the platform decided to re-launch and expand by recruiting a series of trained and compensated volunteers from a variety of communities that they hoped to engage. During the second launch, the community members were able to learn about the platform, and adapt it to their specific use cases. The platform began to gain popularity during a teachers strike in the state of Jharkhand where a specific use case for expressing grievances powered by the community arose.

4.1.3 CAIS

The CAIS platform launched in direct collaboration with the NGO partner for the village every available farmer registered their name, number, and crop type with the NGO partner and consequently the target population was integrated from the start. As the programs grew, each engaged with the community on a high level. In the case of all three platforms, a specific population, and a very specific understanding of that populations needs had to be identified before the platform could be relatively effective. Even after the deployment of the platform, care and close integration with community systems had to be done. The village in which CAIS worked had a system of self-empowerment groups, that had been organized by the NGO. Each group had a leader; and while not every villager in each group had a basic phone, every group leader did. Consequently, the researchers behind CAIS worked to ensure that every group leader was engaged in with CAIS and that they would relay their CAIS informational updates to the villagers that they lead. Similarly, the CAIS researchers worked closely with village leaders to determine who was able to access the information in SMS form and deployed informational physical posters as a substitute to those portions of the village population who could not. This intensive work led to the successful adaptation of technology to the benefit of the farmers yields. The researchers behind the Vaani system also continued to expand the system through a local network of volunteers. (Moitra et al., 2016)

4.1.4 CGNET Swara

CGNet Swara, which we introduced earlier in this paper, also increased their initial participation by engaging with the wider community by holding in-person training and awareness sessions. They have conducted over 50 workshops, and have trained more than 2,000 members of various communities (Marathe et al., 2015). Outreach activities such as these also allowed an increased spread of awareness via word-of-mouth. From these examples, it is clear to see that community engagement is the absolute key to spreading technology. Incentives both monetary and situational are a huge way that these platforms were able to engage their initial users. Incentives served to empower individuals to become champions of the platform and increased the enabled them to use their knowledge of the community and existing peer networks to deliver

the technology where it was needed. All platforms used incentives of some sort; Learn2Earn used a direct payment for recruitment + participation, and also delivered relevant topics to the users. Mobile Vaani provided financial incentives to the volunteers who mobilized to evangelize the product. CAIS did not provide monetary incentives but instead brought technology that had an actionable and tangential impact on the daily lives of farmers. With the deployment of these technologies, direct needs of the population were solved.

5 Conclusion

The boost in recent advancements in NLP research has started breaking down communication and information barriers. This, coupled with in-depth studies on the socio-economic benefits of enabling less-connected communities with technology, provides a strong argument for increasing investment in this area. It is promising to observe increased innovation and steady progress in the empowerment of rural communities using language tools. Increased exposure to the challenges and works in this area can catalyse developments in improving inclusion and information dissemination. We hope that this paper will provide pointers in the right direction for potential ventures that plan on introducing disruptive language technologies to marginalized communities.

References

Stéphane Boyera. 2007. The mobile web to bridge the digital divide. Citeseer.

Kurt Brauchli, Don O'Mahony, L Banach, and M Oberholzer. 2005. ipath-a telemedicine platform to support health providers in low resource settings. *Studies in health technology and informatics*, 114:11–7.

Manu Chopra, Indrani Medhi Thies, Joyojeet Pal, Colin Scott, Bill Thies, and Vivek Seshadri. 2019. Exploring crowdsourced work in low-resource settings. In *ACM Conference on Human Factors in Computing Systems (CHI)*.

Lars Rune Christensen, Hasib Ahsan, Mamunur Rashid, and Badal Kumar Das. 2019. Are you magicians?: The collaborative work of an agricultural information service. In *Proceedings of the Tenth International Conference on Information and Communication Technologies and Development*, ICTD '19, pages 19:1–19:10, New York, NY, USA. ACM.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. 2018. Deep learning for classical japanese literature. *CoRR*, abs/1812.01718.

Sebastien Cuendet, Indrani Medhi, Kalika Bali, and Edward Cutrell. 2013. Videokheti: making video content accessible to low-literate and novice users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2833–2842. ACM.

BK Ganguly and Kalpana Chaudhary. 2003. Forest products of bastar: A story of tribal exploitation. *Economic and Political Weekly*, pages 2985–2989.

Office of the Registrar General & Census Commissioner India. 2011. *Census of India, 2011*. Government of India.

Arun Kumar, Sheetal K Agarwal, and Priyanka Manwani. 2010. The spoken web application framework: user generated content and service creation through low-end mobiles. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, page 2. ACM.

Meghana Marathe, Jacki O'Neill, Paromita Pain, and William Thies. 2015. Revisiting cgnet swara and its impact in rural india. In *Proceedings of the Seventh International Conference on Information and Communication Technologies and Development*, page 21. ACM.

Paul McNamee and James Mayfield. 2002. Comparing cross-language query expansion techniques by degrading translation resources. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 159–166. ACM.

Aparna Moitra, Vishnupriya Das, Gram Vaani, Archna Kumar, and Aaditeshwar Seth. 2016. Design lessons from creating a mobile-based community media platform in rural india. In *Proceedings of the Eighth International Conference on Information and Communication Technologies and Development*, page 14. ACM.

Dripta Piplai Mondal. 2019. Transitional indo-european groups and the use of mundari in selected areas of west bengal. In *Proceedings of The 7th International Conference on Endangered and Lesser Known Languages*.

Christopher Moseley. 2010. *Atlas of the World's Languages in Danger*. Unesco.

Preeti Mudliar, Jonathan Donner, and William Thies. 2013. Emergent practices around cgnet swara: A voice forum for citizen journalism in rural india. *Information Technologies & International Development*, 9(2):pp–65.

U Nations. 2014. United nations e-government survey 2014: E-government for the future we want. *United Nations Department of economic and social affairs*.

Neil Patel, Deepti Chittamuru, Anupam Jain, Paresh Dave, and Tapan S Parikh. 2010. Avaaj otalo: a field study of an interactive voice forum for small farmers in rural india. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 733–742. ACM.

Fang Qiao, Jahanzeb Sherwani, and Roni Rosenfeld. 2010. Small-vocabulary speech recognition for resource-scarce languages. In *Proceedings of the First ACM Symposium on Computing for Development*, page 3. ACM.

K. Ramamritham, A. Bahuman, S. Duttagupta, C. Bahuman, and S. Balasundaram. 2006. Innovative ict tools for information provision in agricultural extension (december 2005). In *2006 International Conference on Information and Communication Technologies and Development*, pages 34–38.

Sebastian Ruder. 2019. The 4 biggest open problems in nlp.

Gary F Simons and Charles D Fennig. 2017. *Ethnologue: Languages of Asia*. sil International.

Stephanie Strassel and Jennifer Tracey. 2016. Lorelei language packs: Data, tools, and resources for technology development in low resource languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3273–3280.

S. Surana, R. Patra, S. Nedevschi, and E. Brewer. 2008. Deploying a rural wireless telemedicine system: Experiences in sustainability. *Computer*, 41(6):48–56.

Saiganesh Swaminathan, Indrani Medhi Thies, Devansh Mehta, Edward Cutrell, Amit Sharma, and William Thies. Learn2earn: Using mobile airtime incentives to bolster public awareness campaigns.

Indrani Medhi Thies et al. 2015. User interface design for low-literate and novice users: Past, present and future. *Foundations and Trends® in Human–Computer Interaction*, 8(1):1–72.

Xiaojun Wan. 2008. Using bilingual knowledge and ensemble techniques for unsupervised chinese sentiment analysis. In *Proceedings of the conference on empirical methods in natural language processing*, pages 553–561. Association for Computational Linguistics.

Dong Zhou, Mark Truran, Tim Brailsford, Vincent Wade, and Helen Ashman. 2012. Translation techniques in cross-language information retrieval. *ACM Computing Surveys (CSUR)*, 45(1):1.

DRCoVe: An Augmented Word Representation Approach using Distributional and Relational Context

Md. Aslam Parwez
Dept. of Computer Sc.
Jamia Millia Islamia
New Delhi, India
aslamparwez.jmi@gmail.com

Muhammad Abulaish
Dept. of Computer Sc.
South Asian University
New Delhi, India
abulaish@sau.ac.in

Mohd. Fazil
Dept. of Computer Sc.
South Asian University
New Delhi, India
mohdfazil.jmi@gmail.com

Abstract

Word representation using the distributional information of words from a sizeable corpus is considered efficacious in many natural language processing and text mining applications. However, distributional representation of a word is unable to capture distant relational knowledge, representing the relational semantics. In this paper, we propose a novel word representation approach using distributional and relational contexts, DRCoVe, which augments the distributional representation of a word using the relational semantics extracted as syntactic and semantic association among entities from the underlying corpus. Unlike existing approaches that use external knowledge bases representing the relational semantics for enhanced word representation, DRCoVe uses typed dependencies (aka syntactic dependencies) to extract relational knowledge from the underlying corpus. The proposed approach is applied over a biomedical text corpus to learn word representation and compared with GloVe, which is one of the most popular word embedding approaches. The evaluation results on various benchmark datasets for *word similarity* and *word categorization* tasks demonstrate the effectiveness of DRCoVe over the GloVe.

1 Introduction

Understanding contextual semantics of words is crucial in many natural language processing (NLP) applications. Recent trends in text mining and NLP suggest immense interest towards learning word embedding or word representation in a vector space from a large corpus, which could be useful for a variety of applications like text classification (Lai et al., 2015), clustering (Wang et al., 2015), and sentiment analysis (Tang et al., 2014). In addition, researchers are devising methods to learn phrase-, sentence-, or document-level embeddings for various NLP applications. Word embeddings capture implicit semantics and hence attracted many researchers to explore and exploit a tremendous amount of available unstructured corpora for efficient word representation by employing mainly unsupervised learning approaches. Further, the growth and availability of domain-specific massive text corpora can be exploited to learn domain-specific word representation.

Although different approaches for learning word embeddings have been proposed in the prior works, they are mostly based on distributional representation of words, considering the neighbors of a word within a fixed context window. These algorithms map sparse representation of words to a lower dimensional vector space where words with similar context appear nearby each other. However, distributional representation of words learned by these algorithms suffer from two important limitations – (i) unable to capture the relational semantics of rare co-occurring words within the corpus, and (ii) unable to capture the relational semantics of words that are outside the purview of the context window. The first limitation is that a large corpus, though represents different contextual information, may have rare co-occurrence of two words because it might not be large enough to possess sufficient count of the co-occurrence of semantically similar word pairs. To overcome this limitation, researchers have incorporated knowledge into these distributional word representations from external knowledge bases (KBs). In this direction, semantically related words in terms of relations like *synonymy*, *hypernymy*, and *meronymy* from KBs like WordNet (Miller, 1995), Freebase (Bollacker et al., 2008) have been used to learn better representation of words (Alsuhaibani et al.,

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 220–229
Hyderabad, India, December 2019.

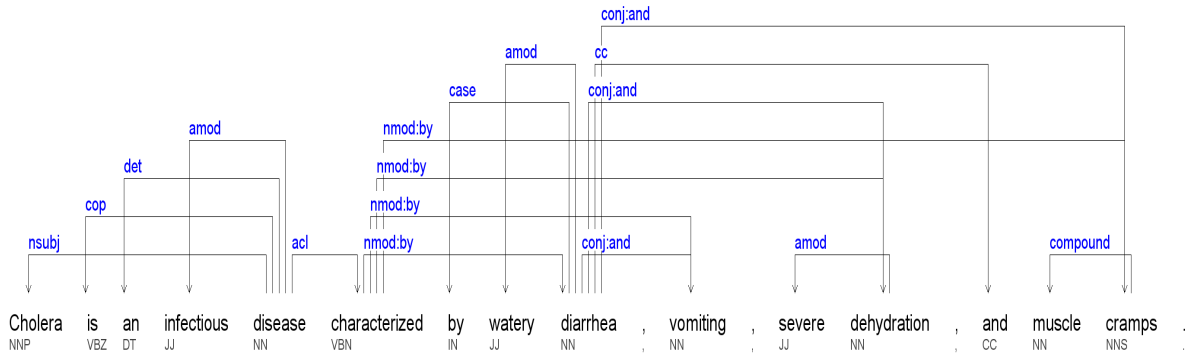


Figure 1: An exemplar dependency parse tree generated by the Stanford parser using DependenSee 3.7.0

2018; Celikyilmaz et al., 2015). This makes these approaches dependent upon the external KBs to enhance the efficacy of word representation. Although KBs provide significant information about word relations, they are scanty with limited entries for each word and does not represent any contextual information. In addition, since KBs are manually curated and maintained, they are not comprehensive.

The second limitation is that the distributional word representations are unable to capture the relational semantics of words due to their dependence on the fixed context window, and hence ignore the semantic associations between words that are outside the purview of the context window. For example, in the sentence, *"cholera is an infectious disease characterized by watery diarrhea, vomiting, severe dehydration, and muscle cramps"*, the word pairs (*cholera*, *dehydration*) and (*cholera*, *cramps*) have long-range dependency. However, both *dehydration* and *cramps* are semantically associated with *cholera* as they are its symptoms. In case of fixed context window size, e.g. 5, such long range dependency relationships will not be captured. Further, if we increase the size of the context window, it will adversely impact the embedding representation due to the inclusion of irrelevant and weak contextual words. Additionally, in case of domain-specific corpus for learning word embedding, the semantic relation between *cholera* and *dehydration*, or *cholera* and *cramps* would be very vital because *dehydration* and *muscle cramps* are the symptoms of *cholera*. These relational semantics can be captured by dependency grammar that shows syntactic and semantic relationships between words of a sentence. To this end, Levy and Goldberg (2014a) presented a dependency-based word representation learning approach to incorporate the *syntactic contexts* instead of *linear contexts*. However, existing literatures have no approach that learn word representation using syntactic contexts extracted from inter-relationships of words based on the dependency tuples generated by the language-parser. For example, in figure 1, the syntactic contexts using only the head and modifier words of the dependency tuples generated by the parser shows direct dependency relation between *cholera* and *disease* through *nsubj* dependency relation; but, it doesn't not show any relational semantics between *cholera* and *watery*, *diarrhea*, *vomiting*, *dehydration*, and *cramps* as they are not directly linked to *cholera* by any dependency relations. Therefore, extraction of such relations to augment word representations would be very helpful for various domain-specific NLP tasks such as classification of disease-related documents or texts. To the best of our knowledge, in the existing literatures, no such approach exists that utilizes the relational semantics extracted from a large corpus to enhance the distributional representation of words.

In this paper, we present an augmented approach, DRCoVe, to use both text corpus and an extracted repository of semantically related triplets from the corpus to learn efficient word representation. The proposed approach first initializes the word representation to low-dimensional real-valued vectors generated from the singular value decomposition (SVD) of positive pointwise mutual information (PPMI) matrix of the underlying corpus and the relational semantic repository. The initial word vectors from the corpus are augmented using vectors from the relational semantic repository, provided the words from the corpus occur in the vocabulary of the relational

semantic repository. In the proposed approach, we implement a modified GloVe (Pennington et al., 2014) objective function for cost optimization to incorporate vector representations from the relational knowledge repository with the initial vectors from the corpus. In brief, the main contributions of this paper can be summarized as follows.

- We propose DRCoVe, a novel approach of learning and augmentation of word representation from a corpus that can handle both long- and short-range dependencies among words.
- The model combines the benefits of point-wise mutual information, singular value decomposition, and neural network-based updation.
- Compared to existing approaches, the proposed model performs considerably better on different benchmark datasets.

Rest of the paper is organized as follows. Section 2 presents a brief review of the existing works on learning word representations. Section 3 presents background details of the concepts used in this paper. Section 4 presents the detailed description of the proposed model. Section 5 presents the experimental details and evaluation results. Finally, section 6 concludes the paper and provides future directions of research.

2 Related Works

Recently, a number of different learning algorithms have been proposed to learn the low-dimensional dense representation of words generally called word embedding used in different NLP tasks such as named entity recognition (Collobert et al., 2011), sentiment analysis (Tang et al., 2014). In this regard, two popular word representation models: continuous bag of words (CBOW) and skip gram (SG) (Mikolov et al., 2013a) models based on neural networks have gained momentum in learning distributed word representation by exploiting the local context of words co-occurring within a given context window. The CBOW predicts the target word given the surrounding context words while SG predicts the surrounding context words given the current word. Similarly, GloVe (Pennington et al., 2014) is another popular method of learning word representation based on global co-occurrence matrix that predicts global co-occurrence between target and context words by employing randomly initialized vectors of desired dimensions. These models learn embeddings only from the corpus without incorporation of any external knowledge. However, in this direction, numerous studies (Yu and Dredze, 2014; Xu et al., 2014; Alsuhaibani et al., 2018) have attempted to incorporate the relational information from KBs for word representation. In Yu and Dredze (2014), the authors proposed an approach to jointly learn embeddings from a corpus and a similarity lexicon (synonymy) by assigning high probabilities to words that appear in the similarity lexicon using joint objective functions of relation constraint models (RCM) and CBOW. Similarly, Xu et al. (2014) used the relational and categorical information as regularization parameters to the SG training objective function to improve the word representation. The CBOW based models normalize target word probabilities for the whole vocabulary, hence, computationally very expensive for large corpora.

In Ghosh et al. (2016), the authors proposed vocabulary driven skip-gram with negative sampling (SGNS) to learn disease-specific word vectors from health-related news corpus by incorporating disease-related vocabulary. Most of the proposed word representation approaches are based on either of the two models (CBOW or SG), or their variants (SGNS, SGHS) of Word2Vec algorithm either by linearly combining additional objective functions or adding as regularizers. Alsuhaibani et al. (2018) used WordNet to extract eight different types of relations such as *synonymy*, *antonymy*, *hypernymy*, *meronymy*, and so on to learn joint embeddings. They used a linear combination of GloVe and KB-based objective functions. All the discussed and existing approaches ignore the relational semantics between the words, which are outside the purview of context-window.

3 Background and Problem Definition

This section presents the notations and the background details of the important concepts used in the proposed approach.

Notations: Suppose a corpus $\mathcal{C}$ has n number of documents $d_1, d_2, \ldots, d_n$, and D represents the collection of target and context words pairs (w, c) obtained from $\mathcal{C}$ for a given context

window size l, where context words of a target word w_i are the surrounding words $w_{i-l}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+l}$. In addition, assume that V_w and V_c represent word and context vocabularies respectively for corpus D. We assume that $n_{(w,c)}$ represents the total count of (w, c) pair in D such that the target word w and context word c appear together within the context window l, n_w and n_c denote the occurrence of w and c respectively in D such that $n_w = \sum_{\hat{c} \in V_c} n_{(w,\hat{c})}$ and $n_c = \sum_{\hat{w} \in V_w} n_{(\hat{w},c)}$. The association between every pair of target and context words of V_w and V_c is presented in a matrix M such that each row of the matrix represents the vector of a target word $w \in V_w$ and each column represents vector of a context word $c \in V_c$ and every element $M_{i,j}$ represents the association between the i^{th} target word w_i and j^{th} context word c_j. Further, assume a relational semantic repository R_l consisting of all the relational semantic triplets extracted from the corpus $\mathcal{C}$. In addition, assume $\mathcal{V}$ represents the vocabulary of R_l. In the paper, alphabets w and c in bold typeface represent vectors.

GloVe: It is a neural network-based machine learning algorithm to learn an efficient lower dimensional dense representation of words in an embedding space. It uses global co-occurrence matrix to learn distributed representation of words from a text corpus. Initially, it creates co-occurrence matrix M with rows representing target words for which we want to learn word representation and the columns represent the context words co-occurring with the target words in the corpus within a given context window. In M, each entry, say, $M_{i,j}$ represents the sum of the reciprocal of the distance of co-occurring target and context words. `GloVe` implements weighted least square regression objective function to minimize the loss J_g as given in equation 1, where $f(M_{w,c})$ is the weight function to find weight between a target word w and context word c as given in equation 2, and b_w and b_c are the bias terms for the underlying target and context words respectively. In the equation 2, $\alpha = 0.75$ is a hyper-parameter and $x_{max} = 100$. The objective of `GloVe` is to minimize the squared difference between the inner product of word and context vectors **w** and **c**, and the logarithm of their co-occurrence count in D.

$$J_g = \frac{1}{2} \sum_{w \in V_w} \sum_{c \in V_c} f(M_{w,c})(\mathbf{w}^T \cdot \mathbf{c} + b_w + b_c - log(M_{i,j}))^2 \tag{1}$$

$$f(M_{w,c}) = min\left\{(M_{w,c}/x_{max})^{\alpha}, 1\right\} \tag{2}$$

In `GloVe`, learning process starts by assigning random vectors of desired dimensions to the target and context words and then updating them during the learning process with an objective to reduce the weighted least square loss as given in equation 1.

Pointwise Mutual Information: In the existing literature, researchers have used different metrics such as co-occurrence count in `GloVe` to represent the association between a word and context pair (w, c). However, simple frequency count is not the best measure of association as it does not incorporate any contextual information. The pointwise mutual information (PMI) is another measure of association and better as compared to co-occurrence count. It measures how often two events co-occur compared to what we would expect if they were independent as defined in equation 3 (Jurafsky and Martin, 2018). There can be target and context word pairs ($w \in V_w$ and $c \in V_c$) which do not appear together within the given context window l in the corpus and for such pairs $n_{(w,c)} = 0$, and therefore $PMI(w, c) = log(0) = -\infty$. To avoid this situation, positive pointwise mutual information (PPMI) has been used in which negative `PMI` values are mapped to zero as given in equation 4. In addition, Bullinaria and Levy (2007) showed that `PPMI` performs better than `PMI` in finding semantic similarity. `PPMI` measures are widely used to find semantic similarity, however, these matrices are highly sparse and need huge computational resources. One measure is to convert such sparse vectors into low dimensional dense vectors to improve computational efficiency and generalization. In this regard, dimensionality reduction is a way to find low dimensional dense vectors using matrix factorization techniques such as `SVD`.

$$PMI(w, c) = log\left(\frac{P(w, c)}{P(w) * P(c)}\right) = log\left(\frac{n_{(w,c)} * |D|}{n_w * n_c}\right) \tag{3}$$

$$PPMI(w, c) = max\left\{PMI(w, c), 0\right\} \tag{4}$$

Singular Value Decomposition: It is a dimensionality reduction method which decomposes a symmetric matrix $M_{m\times n}$ into three matrices U, Σ, and V such that $M = U \cdot \Sigma \cdot V$. The matrices U and V are orthogonal matrices while Σ is a diagonal matrix of singular values. To obtain d dimensional vectors, the matrix M is decomposed to $U_{m\times d}$, $\Sigma_{d\times d}$, and $V_{d\times n}$ corresponding to top d singular values. The d-dimensional rows of matrix $W = U \cdot \sqrt{\Sigma}$ are dense vectors which are the approximate representative of high dimensional rows of M. The matrix W is considered as a dense vector representation of words, while the matrix $C = V^T \cdot \sqrt{\Sigma}$ can be considered as context representation. The matrices W and C thus obtained are used as initial word and context representations respectively. These resulting representations need to fulfill minimization of error in matrix decomposition.

4 Proposed Approach

This section presents the detailed description of the proposed approach, starting from the mechanism to generate initial word representation from the corpus, their augmentation through relational semantics, and finally, adaptive updation of word vectors. A detailed description of each step of the proposed approach is presented in the following subsections.

4.1 Initial Vector Representation

To learn word representation of desired dimension, we first need to initialize the vectors for each target and context words pair of the corpus that can be further augmented using their relational semantics and updated based on the weighted least square loss minimization process. Before neural-based approaches, distributed word representations were based on count-based vectors such as *tf-idf* and *SVD-based* vectors. Recent advancements in neural network-based word representation have shown significant improvement in its performance in various NLP tasks. The neural network-based word representations are based on prediction (Mikolov et al., 2013b,a) of either the target word given the context within the specified context window or vice versa. However, recent studies (Levy and Goldberg, 2014b; Levy et al., 2015) have shown that the neural network-based embedding learned using Word2Vec or GloVe models are comparable in performance with the traditional representation of vectors obtained through the decomposition of PPMI matrix. Therefore, to incorporate the benefits of traditional decomposition-based vectors, the proposed approach generates initial word representation using vectors obtained from SVD-based factorization of PPMI matrix. To this end, we first create a co-occurrence matrix M considering the co-occurrence count of every (w, c) pair of target and context words from V_w and V_c respectively that is further mapped to a PPMI matrix M_p. Thereafter, the M_p is factorized using SVD to generate initial low dimensional dense vector representations of target and context words as $W = U \cdot \sqrt{\Sigma}$ and $C = V^T \cdot \sqrt{\Sigma}$, respectively from the corpus that incorporate the distributional semantics. Similarly, the same process is repeated for relational semantic repository R_l to generate the initial vector representation of target and context words as $\hat{W} = U \cdot \sqrt{\Sigma}$ and $\hat{C} = V^T \cdot \sqrt{\Sigma}$, respectively from R_l. The initial vectors of target and context words from the corpus are further augmented using the vectors generated from the relational semantic repository. A detailed description of the augmentation process is described in the following section.

4.2 Objective Function Augmentation

To minimize the decomposition error we followed the GloVe approach of optimization of the initial representation of vectors. GloVe method learns continuous word representation from a corpus using the global co-occurrence matrix. However, Glove does not incorporate any additional or domain-specific knowledge and suffers from two important limitations as discussed in section 1. Therefore, during optimization we performed the augmentation of initial word representation from the corpus by merging with the initial word representation from the relational semantic repository. To augment the additional information during learning, we define an augmented objective function J_a similar to GloVe as given in equation 5, where $f(p_{w,c})$ is the weight function to assign weight between every pair (w, c) of target and context words as given in equation 6, and b_w and b_c are the bias values for w and c respectively, and $p_{w,c}$ is the PPMI value between w and c. In equation 6, α is a hyper parameter and we used

0.75 as its value as used in GloVe.

$$J_a = \frac{1}{2} \sum_{w \in V_w} \sum_{c \in V_c} f(p_{w,c})(\mathbf{w'}^T \cdot \mathbf{c'} + b_w + b_c - log(p_{w,c}))^2 \quad (5)$$

$$f(p_{w,c}) = min \left\{ \left(p_{w,c} / \max_{\forall w,c \in D} (p_{w,c}) \right)^{\alpha}, 1 \right\} \quad (6)$$

Thereafter, we employed the relational semantics from the extracted relational semantic repository R_l consisting of vocabulary $\mathcal{V}$ to augment the learning process. The input corpus $\mathcal{C}$ consist of target and context words pairs $(w, c) \in D$. Based on $\mathcal{V}$, we grouped the (w, c) pairs of D into three categories – $D_\wedge$, $D_\sim$, and $D_\oplus$ such that

- $D_\wedge = \{(w, c) : w \in \mathcal{V} \wedge c \in \mathcal{V}\}$, i.e. both the target and context words belongs to $\mathcal{V}$
- $D_\sim = \{(w, c) :\sim (w \in \mathcal{V} \wedge c \in \mathcal{V})\}$, i.e. neither target nor the context word belongs to $\mathcal{V}$
- $D_\oplus = \{(w, c) : w \in \mathcal{V} \oplus c \in \mathcal{V}\}$, i.e either the target or the context word belongs to $\mathcal{V}$

We need to consider each of these (w, c) pair categories especially while merging to generate augmented word representation.

In case of $D_\wedge$, as both the target and context words belong to $\mathcal{V}$, we considered the merged vectors from the corpus and the relational semantic repository corresponding to target and context words such that $\mathbf{w'} = 0.5 * (\mathbf{w} + \hat{\mathbf{w}})$ and $\mathbf{c'} = 0.5 * (\mathbf{c} + \hat{\mathbf{c}})$, where $\mathbf{w}$ and $\mathbf{c}$ are the initial vectors from corpus and $\hat{\mathbf{w}}$ and $\hat{\mathbf{c}}$ are the initial vectors from relational semantic repository. For category $D_\sim$, we considered the initial vectors from the corpus only as neither of the two words belongs to $\mathcal{V}$, hence, we have $\mathbf{w'} = \mathbf{w}$ and $\mathbf{c'} = \mathbf{c}$. Similarly, in case of $D_\oplus$, as either of the two words belongs to $\mathcal{V}$ but not both, we took the merged vector for target or context word depending upon which word belongs to $\mathcal{V}$. In this case, if target word belongs to $\mathcal{V}$, we take $\mathbf{w'} = 0.5 * (\mathbf{w} + \hat{\mathbf{w}})$ and if context word belongs to $\mathcal{V}$, we consider $\mathbf{c'} = 0.5 * (\mathbf{c} + \hat{\mathbf{c}})$.

4.3 Adaptive Updation of Parameters

We performed the parameter updation during learning process based on a well-known gradient descent technique called AdaGrad (Duchi et al., 2011), which is an adaptive gradient update algorithm to perform gradient-based learning. The gradients are computed as follows:

Table 1: Concept categorization performance with $l = 5$, and $d = 100$

Word Embeddings	AP	BLESS	Battig	ESSLI_1a	ESSLI_2b	ESSLI_2c
GloVe_W	0.1940	0.21	0.0999	0.4090	**0.575**	0.3333
GloVe_Merged	**0.2213**	0.21	0.1062	0.4318	0.55	0.3555
DRCoVe_W	0.1890	0.235	0.0995	0.4318	0.45	0.377
DRCoVe_C	0.1990	**0.26**	0.1062	**0.4772**	0.475	**0.4222**
DRCoVe_Merged	0.1965	0.245	**0.1081**	0.4545	0.5	0.4

Table 2: Concept categorization performance with $l = 5$, and $d = 200$

Word Embeddings	AP	BLESS	Battig	ESSLI_1a	ESSLI_2b	ESSLI_2c
GloVe_W	0.1815	0.205	0.0982	0.4318	**0.55**	0.3777
GloVe_Merged	0.2039	0.225	0.1049	**0.4545**	0.525	0.3777
DRCoVe_W	0.1940	**0.23**	0.1013	0.4090	0.475	0.3777
DRCoVe_C	0.2064	0.215	**0.1060**	0.4090	0.475	0.3777
DRCoVe_Merged	**0.2068**	0.225	0.1009	0.4318	0.45	**0.4**

$$\frac{\delta J}{\delta \mathbf{w'}} = g_{t,\mathbf{w'}} = \sum_{c \in V_c} f(p_{w,c})(\mathbf{w'}^T \cdot \mathbf{c'} + b_w + b_c - log(p_{w,c})) \cdot \mathbf{c'} \quad (7)$$

$$\frac{\delta J}{\delta \mathbf{c'}} = g_{t,\mathbf{c'}} = \sum_{w \in V_w} f(p_{w,c})(\mathbf{w'}^T \cdot \mathbf{c'} + b_w + b_c - log(p_{w,c})) \cdot \mathbf{w'} \quad (8)$$

$$\frac{\delta J}{\delta b_w} = g_{t,b_w} = \sum_{c \in V_c} f(p_{w,c})(\mathbf{w'}^T \cdot \mathbf{c'} + b_w + b_c - log(p_{w,c})) \quad (9)$$

$$\frac{\delta J}{\delta b_c} = g_{t,b_c} = \sum_{w \in V_w} f(p_{w,c})(\mathbf{w'}^T \cdot \mathbf{c'} + b_w + b_c - log(p_{w,c})) \quad (10)$$

AdaGrad algorithm is suitable for dealing with sparse data as it performs larger updates for infrequent words, and smaller updates for frequent words. The update equation is shown as follows:

$$\mathbf{w'}^{t+1} = \mathbf{w'}^t - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,\mathbf{w'}}^2}} * (g_{t,\mathbf{w'}}) \quad (11)$$

where, $\mathbf{w'}$ is the a merged target word vector, $g_{t,w}$ is the gradient at time t, and $g_{\tau,w}^2$ is the squared gradient at time τ for the target word vector $\mathbf{w'}$. Similarly, updates for context word and biases are performed according to the following equations.

$$\mathbf{c'}^{t+1} = \mathbf{c'}^t - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,\mathbf{c'}}^2}} * (g_{t,\mathbf{c'}}) \quad (12)$$

$$b_w^{t+1} = b_w^t - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,b_w}^2}} * (g_{t,b_w}) \quad (13)$$

$$b_c^{t+1} = b_c^t - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,b_c}^2}} * (g_{t,b_c}) \quad (14)$$

Table 3: Concept categorization performance with $l = 10$, and $d = 100$

Word Embeddings	AP	BLESS	Battig	ESSLI_1a	ESSLI_2b	ESSLI_2c
GloVe_W	0.204	0.215	0.1032	0.4091	0.525	0.3778
GloVe_Merged	0.2113	0.22	0.1095	0.4308	**0.525**	0.3778
DRCoVe_W	0.2015	**0.25**	0.0996	0.4091	0.475	0.3778
DRCoVe_C	**0.2139**	0.22	0.1017	**0.4318**	0.475	**0.4222**
DRCoVe_Merged	0.199	0.225	**0.1047**	0.4091	0.45	0.3556

Table 4: Concept categorization performance with $l = 10$, and $d = 200$

Word Embedding	AP	BLESS	Battig	ESSLI_1a	ESSLI_2b	ESSLI_2c
GloVe_W	0.1965	0.2150	0.1076	0.4090	0.55	0.4222
GloVe_Merged	**0.2313**	0.22	0.1106	0.4140	**0.625**	0.4
DRCoVe_W	0.2064	0.225	0.1026	**0.4545**	0.475	0.355
DRCoVe_C	0.1965	**0.23**	0.1085	0.4014	0.525	0.4
DRCoVe_Merged	0.2114	0.225	**0.1122**	0.4245	0.45	**0.432**

5 Experimental Setup and Results

The DRCoVe is evaluated on different benchmark datasets using two evaluation tasks – *word similarity* and *concept categorization*. This section presents a brief description of corpus and relational semantic repository used in the evaluation process, experimental setup, and finally presents the evaluation results.

5.1 Corpus and Relational Semantic Repository

The DRCoVe is evaluated on a biomedical text corpus crawled from PubMed[1], an online repository of millions of citations and abstracts related to *biomedicine*, *health*, *life and behavioral sciences*, and *bioengineering*. The abstracts are the source of rich information related to *diseases*, *symptoms*, *pathogens*, *vectors*, and their *transmission* and *etiologies*. PubMed provides access to the abstracts of documents through *axis 2.1.6.2 API*[2]. The crawled corpus $\mathcal{C}$ consist of 16,337 PubMed documents related to four diseases – *cholera*, *dengue*, *influenza*, and *malaria*. In addition, a relational semantic repository R_l is created by extracting relational triplets $< arg_1,\ relation,\ arg_2 >$ based on typed dependencies generated by Stanford parser[3] that are filtered using MetaMap[4] to identify meaningful disease-symptom triplets. The repository R_l is used to augment the learning process of word representation. We have extracted the association between the diseases and symptoms using the approach defined in (Parwez et al., 2018; Abulaish et al., 2019).

5.2 Experimental Setup

The documents of the corpus $\mathcal{C}$ are tokenized and processed by removing numbers, punctuations, and stop words. We experimented with the context window size l of 5 and 10 (i.e. for $l = 5$, the context words are the 5 preceding and 5 succeeding words to the target word) to extract the context words from the corpus. The co-occurrence matrix is created using the co-occurrence frequencies of the target and context words pair within the corpus. The co-occurrence matrix is further mapped into PPMI matrix, which is further factorized using SVD to get the initial word vector of desired dimension $d \in \{100, 200\}$. A similar procedure is repeated for relational semantic repository R_l and initial vectors are generated for the target and context words. Thereafter, initial word representation of corpus is augmented using the word representation of relational semantic repository which is then optimized using the objective function defined in equation 5. We used a stochastic gradient-based algorithm *AdaGrad* with the learning rate $\eta = 0.05$ for optimization. The proposed algorithm is executed for 50 iterations to converge into an optimum solution. As a result, we obtain two sets of enhanced embeddings, one for the target words of vocabulary V_w and another for the context words of vocabulary V_c. It has been shown that when the two embeddings of a word are combined by taking an average of the corresponding word vectors, the resultant embedding performs better (Pennington et al., 2014). We have presented results for both the word and context representation in addition to their merged representation.

5.3 Evaluation Results and Comparative Analysis

The quality of the learned word vectors based on *DRCoVe* is evaluated using *concept categorization* and *similarity prediction* tasks.

Concept Categorization: We evaluated the quality of learned word embedding based on *concept categorization*. It is the grouping of concepts from a given set of concepts into different categories. It evaluates the word representation by clustering the learned vectors into different groups. The performance is assessed based on the extent to which each cluster possesses concepts from a given category. The evaluation metric is

[1] https://www.ncbi.nlm.nih.gov/pubmed/
[2] http://axis.apache.org/axis2/java/core/
[3] http://nlp.stanford.edu/software/lex-parser.shtm
[4] https://metamap.nlm.nih.gov/

Table 5: Word similarity performance with $l = 5$, and $d = 100$

Word Embeddings	MTurk	RG65	RW	SCWS	SimLex999	TR9856	WS353	WS353R	WS353S
GloVe_W	0.1869	-0.0650	0.1881	0.27104	0.0407	0.1259	0.2288	0.1411	0.2404
GloVe_Merged	0.1976	-0.0675	**0.1891**	0.2844	0.0354	0.1275	0.2269	0.1447	0.2300
DRCoVe_W	**0.2327**	0.1726	0.1513	0.29	0.0737	0.1347	0.2881	0.2338	0.2702
DRCoVe_C	0.2049	0.1368	0.1555	0.2964	0.0780	**0.1454**	**0.2961**	**0.2467**	**0.2762**
DRCoVe_Merged	0.2270	**0.1839**	0.1284	**0.2982**	**0.0907**	0.1382	0.2690	0.2458	0.2324

Table 6: Word similarity performance with $l = 5$, and $d = 200$

Word Embeddings	MTurk	RG65	RW	SCWS	SimLex999	TR9856	WS353	WS353R	WS353S
GloVe_W	0.1915	-0.0430	0.1877	0.2837	0.0383	0.1263	0.2420	0.1542	0.2471
GloVe_Merged	0.2043	-0.0567	**0.1894**	0.2842	0.0316	0.1275	0.2314	0.1462	0.2374
DRCoVe_W	0.1919	0.087	0.1563	**0.3019**	0.0739	**0.1468**	0.2949	0.2260	0.2662
DRCoVe_C	**0.2152**	0.1336	0.1544	0.3007	0.0811	0.1405	**0.3120**	**0.2385**	**0.2966**
DRCoVe_Merged	0.2038	**0.1390**	0.1347	0.2977	**0.0915**	0.1412	0.2833	0.2272	0.250

called *purity* and it is 100% if the given standard category is reproduced completely. On the other hand, *purity* reaches to 0 when cluster quality worsens. The DRCoVe is evaluated based on *concept categorization* using 6 different benchmark datasets: *AP*, *BLESS*, *Battig*, *ESSLI_1a*, *ESSLI_2b*, and *ESSLI_2c*. The evaluation and comparison results on different combination of context window size and dimensionality over 6 benchmark datasets are given in tables 1, 2, 3, and 4 respectively. It can be observed from the tables that for concept categorization task, except *ESSLI_2b*, in most of the cases, DRCoVe embedding performs better than the GloVe embeddings.

Word Similarity: To evaluate learned vectors on *word similarity* task, we computed cosine similarity between learned embedding of word pairs and evaluated it based on average similarity rating assigned by human annotators to these word pairs from the benchmark datasets. The idea here is that the learned embeddings encapsulate semantics of the words if there is greater extent of correlation between the similarity score computed from the learned word vectors and the similarity score assigned by the human annotators. We calculated Spearman's rank correlation coefficient between the cosine similarity of learned embeddings and human rated similarity of word pairs. We used 9 different benchmark datasets – *MTurk*, *RG65*, *RW*, *SCWS*, *SimLex999*, *TR9856*, *WS353*, *WS353R*, and *WS353S* for evaluation. In addition, we also compared the quality of learned representation in terms of similarity task with the two variants of GloVe: GloVe_W and GloVe_Merged. The evaluation and comparison results on different combination of context window size and dimensionality on the benchmark datasets for *word similarity* are given in tables 5, 6, 7, and 8 respectively. On analysis of tables, it can be found that the *context* and *merged* vectors of DRCoVe are significantly better as compared to GloVe word vectors and merged vectors except RW, where GloVe is better.

6 Conclusion and Future Direction

Word embeddings learned from diverse sources using methods like GloVe as the distributional representation of words have been employed to resolve numerous natural language processing problems with considerable accuracy. However, these distributional representations are unable to capture the relational semantics of distant words and the words with rare co-occurrences in the corpus. In this paper, we have proposed DRCoVe, an augmentation approach of distributional word representations from a corpus with relational semantic information extracted from the corpus to learn enhanced word representation. We compared the proposed model based on semantic similarity and concept categorization tasks on different benchmark datasets and found that the word representation learned by DRCoVe shows better performance than the GloVe model in most of the datasets. The learned word representations could be useful for various NLP tasks like text classification or concept categorization. Learning word representations over much larger corpus and evaluation of their efficacy for short texts

Table 7: Word similarity performance with $l = 10$, and $d = 100$

Word Embeddings	MTurk	RG65	RW	SCWS	SimLex999	TR9856	WS353	WS353R	WS353S
GloVe_W	0.2223	-0.0625	0.1852	0.3013	0.0469	0.1341	0.2429	0.1776	0.2795
GloVe_Merged	0.2267	-0.0524	**0.1863**	0.3102	0.0411	0.1351	0.2393	0.1693	0.2774
DRCoVe_W	0.1930	0.0937	0.1637	0.2951	0.0538	**0.1396**	0.3179	0.2304	0.3250
DRCoVe_C	**0.2309**	0.1241	0.1639	0.2989	0.0441	0.1383	**0.3310**	**0.2506**	**0.3336**
DRCoVe_Merged	0.2085	**0.1404**	0.1393	**0.3140**	**0.0599**	0.1372	0.3033	0.2341	0.2760

Table 8: Word similarity performance with $l = 10$, and $d = 200$

Word Embeddings	MTurk	RG65	RW	SCWS	SimLex999	TR9856	WS353	WS353R	WS353S
GloVe_W	0.2209	-0.0759	0.1855	0.2943	0.0401	0.1321	0.2448	0.1755	0.2742
GloVe_Merged	0.2262	-0.0613	**0.1863**	0.3023	0.0351	0.1347	0.2398	0.1659	0.2779
DRCoVe_W	0.2040	0.0890	0.1734	**0.3162**	0.0775	**0.1425**	0.3029	0.2278	0.3101
DRCoVe_C	**0.2377**	**0.1539**	0.1755	0.3125	0.0793	0.1408	**0.3134**	**0.2364**	**0.3236**
DRCoVe_Merged	0.1848	0.1445	0.1244	0.3088	**0.0970**	0.1363	0.2545	0.2230	0.2219

like tweets classification seems one of the future directions of research.

References

Muhammad Abulaish, Md. Aslam Parwez, and Jahiruddin. 2019. Disease: A biomedical text analytics system for disease symptom extraction and characterization. *Journal of Biomedical Informatics*, 100(12):1–15.

Mohammed Alsuhaibani, Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2018. Jointly learning word embeddings using a corpus and a knowledge base. *PloS one*, 13(3):1–26.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of International Conference on Manageent of Data*, pages 1247–1250, Vancouver, Canada. ACM.

John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39(3):510–526.

Asli Celikyilmaz, Dilek Hakkani-Tur, Panupong Pasupat, and Ruhi Sarikaya. 2015. Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems. In *2015 AAAI Spring Symposium Series*, pages 39–42, California, USA. AAAI.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(1):2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(1):2121–2159.

Saurav Ghosh, Prithwish Chakraborty, Emily Cohn, John S Brownstein, and Naren Ramakrishnan. 2016. Characterizing diseases from unstructured text: A vocabulary driven word2vec approach. In *Proceedings of International Conference on Information and Knowledge Management*, pages 1129–1138, Indianapolis, USA. ACM.

Daniel Jurafsky and James H. Martin. 2018. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 3. Prentice-Hall, Inc.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of International Conference on Artificial Intelligence*, pages 2267–2273, Texas, USA. AAAI.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 302–308, Maryland, USA. ACL.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Proceedings of International Conference on Neural Information Processing Systems*, pages 2177–2185, Montreal, Canada. Curran Associates.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

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 S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of International Conference on Neural Information Processing Systems*, pages 3111–3119, Nevada, USA. Curran Associates.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Md Aslam Parwez, Muhammad Abulaish, and Jahiruddin. 2018. Biomedical text analytics for characterizing climate-sensitive disease. *Procedia Computer Science*, 132:1002–1011.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of International Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar. ACL.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1555–1565, Maryland, USA. ACL.

Peng Wang, Jiaming Xu, Bo Xu, Cheng-Lin Liu, Heng Zhang, Fangyuan Wang, and Hongwei Hao. 2015. Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 352–357, Beijing, China. AAAI.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of International Conference on Information and Knowledge Management*, pages 1219–1228, Shanghai, China. ACM.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 545–550, Maryland, USA. ACL.

A Deep Learning Approach for Automatic Detection of Fake News

Tanik Saikh[1] **Arkadipta De**[2] **Asif Ekbal**[1] **Pushpak Bhattacharyya**[1]
Department of Computer Science and Engineering[1]
Indian Institute of Technology Patna[1]
Department of Computer Science and Engineering[2]
Government College of Engineering and Textile Technology, Berhampore[2]
{tanik.srf17, asif, pb}@iitp.ac.in[1]
de.arkadipta05@gmail.com[2]

Abstract

Fake news detection is a very prominent and essential task in the field of journalism. This challenging problem is seen so far in the field of politics, but it could be even more challenging when it is to be determined in the multi-domain platform. In this paper, we propose two effective models based on deep learning for solving fake news detection problem in online news contents of multiple domains. We evaluate our techniques on the two recently released datasets, namely *FakeNews AMT and Celebrity* for fake news detection. The proposed systems yield encouraging performance, outperforming the current handcrafted feature engineering based state-of-the-art system with a significant margin of 3.08% and 9.3% by the two models, respectively. In order to exploit the datasets, available for the related tasks, we perform cross-domain analysis (i.e. model trained on FakeNews AMT and tested on Celebrity and vice versa) to explore the applicability of our systems across the domains.

1 Introduction

In the emergence of social and news media, data are constantly being created day by day. The data so generated are enormous in amount, and often contains miss-information. Hence it is necessary to check it's truthfulness. Nowadays people mostly rely on social media and many other online news feeds as their only platforms for news consumption (Jeffrey and Elisa, 2016). A survey from the *Consumer News and Business Channel (CNBC)* also reveals that more people are rely on social media for news consumption rather than news paper [1]. Therefore, in order to deliver the genuine news to such consumers, checking the truthfulness of such online news content is of utmost priority to news industries. The task is very difficult for a machine as even human being can not understand news article's veracity (easily) after reading the article.

Prior works on fake news detection entirely rely on the datasets having satirical news contents sources, namely *"The Onion" (Rubin et al., 2016)*, fact checking website like *Politi-Fact (Wang, 2017), and Snopes (Popat et al., 2016)*, and on the contents of the websites which track viral news such as *BuzzFeed (Potthast et al., 2018)* etc. But these sources have severe drawbacks and multiple challenges too. Satirical news mimic the real news which are having the mixture of irony and absurdity. Most of the works in fake news detection fall in this line and confine in one domain (i.e. politics). The task could be even more challenging and generic if we study this fake news detection problem in multiple domain scenarios. We endeavour to mitigate this particular problem of fake news detection in multiple domains. This task is even more challenging compared to the situation when news is taken only from a particular domain, i.e. uni-domain platform. We make use of the dataset which contained news contents from multiple domains. The problem definition would be as follows:

Given a *News Topic* along with the corresponding *News Body Document*, the task is to classify whether the given news is *legitimate/genuine* or *Fake*. The work described in Pérez-Rosas et al. (2018) followed this path. They also offered two novel computational resources, namely *FakeNews AMT* and *Celebrity news*. These datasets are having triples of *topic, document and label (Legit/Fake)* from multiple domains (like *Business, Education, Technology, Entertainment and Sports* etc) including *politics*. Also, they claimed that these datasets focus on the deceptive properties of

[1] https://www.cnbc.com/2018/12/10/social-media-more-popular-than-newspapers-for-news-pew.html

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 230–238
Hyderabad, India, December 2019.

online articles from different domains. They provided a baseline model. The model is based on Support Vector Machine (SVM) that exploits the hand-crafted linguistics features. The SVM based model achieved the accuracies of 74% and 76% in the FakeNews AMT and Celebrity news datasets, respectively. We pose this problem as a classification problem. So the proposed predictive models are binary classification systems which aim to classify between fake and the verified content of online news from multiple domains. We solve the problem of multi-domain fake news detection using two variations of deep learning approaches. The first model (denoted as Model 1) is a Bi-directional Gated Recurrent Unit (BiGRU) based deep neural network model, whereas the second model (i.e. Model 2) is *Embedding from Language Model (ELMo)* based. It is to be noted that the use of deep learning to solve this problem in this particular setting is, in itself, very new. The technique, particularly the word attention mechanism, has not been tried for solving such a problem. Existing prior works for this problem mostly employ the methods that make use of handcrafted features. The proposed systems do not depend on hand crafted feature engineering or a sophisticated NLP pipeline, rather it is an end to end deep neural network architecture. Both the models outperform the state-of-the-art system.

2 Related Work

A sufficient number of works could be found in the literature in fake news detection. Nowadays the detection of fake news is a hot area of research and gained much more research interest among the researchers. We could detect fake news at two levels, namely the conceptual level and operational level. Rubin et al. (2015) defined that conceptually there are three types of fake news: *viz i. Serious Fabrications ii. Hoaxes and iii. Satire*. The work of Conroy et al. (2015) fostered linguistics and fact checking based approaches to distinguish between real and fake news, which could be considered as the work at conceptual level. Chen et al. (2015) described that fact-checking approach is a verification of hypothesis made in a news article to judge the truthfulness of a claim. Thorne et al. (2018) introduced a novel dataset for fact-checking and verification where evidence is large Wikipedia corpus. Few notable works which made use of text as evidence can be found in (Ferreira and Vlachos, 2016; Nie et al., 2018).

The Fake News Challenge [2] organized a competition to explore, how artificial intelligence technologies could be fostered to combat fake news. Almost 50 participants were participated and submitted their systems. Hanselowski et al. (2018) performed retrospective analysis of the three best participating systems of the Fake News Challenge. The work of Saikh et al. (2019) detected fake news through stance detection and also correlated this stance classification problem with Textual Entailment (TE). They tackled this problem using statistical machine learning and deep learning approaches separately and with combination of both of these. This system achieved the state of the art result.

Another remarkable work in this line is the verification of a human- generated claim given the whole Wikipedia as evidence. The dataset, namely (*Fact Extraction and Verification (FEVER)*) proposed by Thorne et al. (2018) served this purpose. Few notable works in this line could be found in (Yin and Roth, 2018; Nie et al., 2019).

3 Proposed Methods

We propose two deep Learning based models to address the problem of fake information detection in the multi-domain platform. In the following subsections, we will discuss the methods.

3.1 Model 1

This model comprises of multiple layers as shown in the Figure 1. The layers are *A. Embedding Layer B. Encoding Layer (Bi-GRU) C. Word level Attention D. Multi-layer Perceptron (MLP).*

A. Embedding Layer: The embedding of each word is obtained using pre-trained fastText model[3](Bojanowski et al., 2017). FastText embedding model is an extended version of Word2Vec (Mikolov et al., 2013). Word2Vec (predicts embedding of a word based on given context and vice-versa) and Glove (exploits count and word co-occurrence matrix to predict embedding of a word) (Pennington et al., 2014) both treat each word as an atomic entity. The fastText model produces embedding of each word by combining the embedding of each character n-gram of that word. The model works better on rare words and also produces embedding for

[2] http://www.fakenewschallenge.org/
[3] https://fasttext.cc/

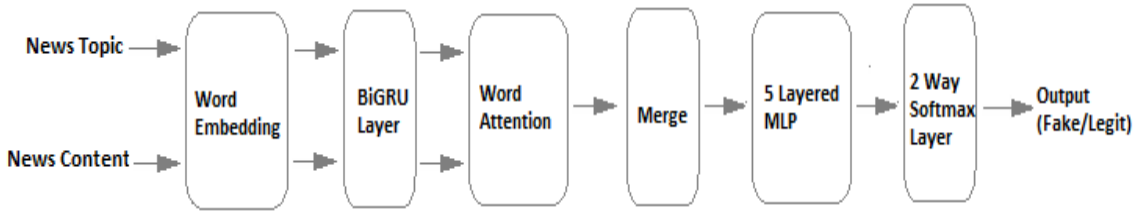


Figure 1: Architectural Diagram of the Proposed First System

out-of-vocabulary words, where Word2Vec and Golve both fail. In the multi-domain scenario vocabularies are from different domains and there is a high chance of existing different domain specific vocabularies. This is the reason for choosing the fastText word vector method.

*B. **Encoding Layer:*** The representation of each word is further given to a bidirectional Gated Recurrent Units (GRUs) (Cho et al., 2014) model. GRU takes less parameter and resources compared to Long Short Term Memory (LSTM), training also is computationally efficient. The working principles of GRU obey the following equations:

$$z = \alpha(x_t U^z + s_{t-1} W^z) \quad (1)$$

$$r = \alpha(x_t U^r + s_{t-1} W^r) \quad (2)$$

$$h = tanh(x_t U^h + r_t \cdot s_{t-1} W^r) \quad (3)$$

$$r = (1 - z) \cdot h + z \cdot s_{t-1} \quad (4)$$

In equation 1, z is the update gate at time step t. This z is the summation of the multiplications of x_t with it's own weight U(z) and s_{t-1} (holds the information of previous state) with it's own W(z). A sigmoid α is applied on the summation to squeeze the result between 0 and 1. The task of this update gate (z) is to help the model to estimate how much of the previous information (from previous time steps) needs to be passed along to the future. In the equation 2, r is the reset gate, which is responsible for taking the decision of how much past information to forget. The calculation is same as the equation 1. The differences are in the weight and gate usages. The equation 3 performs as follows, i. multiply input x_t with a weight U and s_{t-1} with a weight W. ii. Compute the element wise product between reset gate r_t and s_{t-1}W. Then a non-linear activation function *tanh* is applied to the summation of i and ii. Finally, in the equation 4, we compute r which holds the information of the current unit. The computation procedure is as follows: i. compute element-wise multiplication to the update gate z_t and $s_{(t-1)}$. ii. calculate element-wise multiplication to (1-z) with h. Take the summation of i and ii.

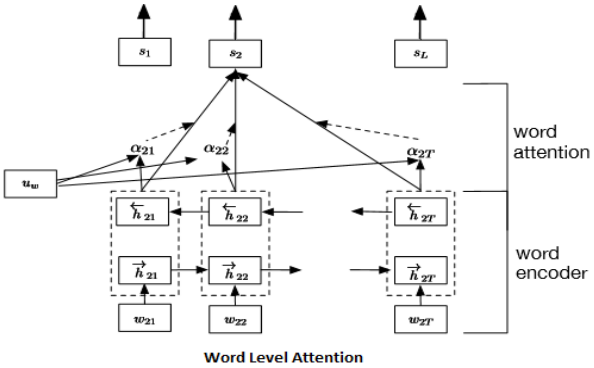


Figure 2: Word Level Attention Network

The bidirectional GRUs consists of the forward GRU, which reads the sentence from the first word (w_1) to the last word (w_L) and the backward GRU, that reads in reverse direction. We concatenate the representation of each word obtained from both the passes.

*C. **Word Level Attention:*** We apply the attention model at word level (Bahdanau et al., 2015; Xu et al., 2015). The objective is to let the model decide which words are importance compared to other words while predicting the target class (fake/legit). We apply this as applied in Yang et al. (2016). The diagram is shown in the Figure 2. We take the aggregation of those words' representation which are multiplied with attention weight to get sentence representation. We do this process for both the news topic and the corresponding document. This particular technique of the word attention mechanism, has not been tried for solving such a problem.

$$U_{i_t} = tanh(W_w h_{i_t} + b_w) \quad (5)$$

$$\alpha_{i_t} = \frac{exp(u_{i_t}^T u_w)}{\sum_t exp(u_{i_t}^T u_w)} \quad (6)$$

$$s_i = \sum_t \alpha_{i_t} h_{i_t} \quad (7)$$

First get the word annotation h_{i_t} through GRU output and compute u_{i_t} as a hidden representation of h_{i_t} in 5. We measure the importance of the word as the similarity of u_{i_t} with a word level context vector u_w and get a normalized importance weight α_{i_t} through a softmax in 6. After that, in 7, we compute the sentence vector s_i as a weighted sum of the word annotations based on the weights α_{i_t}. The word context vector u_w is randomly initialized and jointly learned during the training process.

D. Multi-Layer Perceptron: We concatenate the sentence vector obtained for both the inputs. The obtained vector further fed into fully connected layers. We use 512, 256, 128, 50 and 10 neurons, respectively, for five such layers with ReLU (Glorot et al., 2011) activation in each layer. Between each such layer, we employ 20% dropout (Srivastava et al., 2014) as a measurement of regularization. Finally, the output from the last fully connected layer is fed into a final classification layer with softmax (Duan et al., 2003) activation function having 2 neurons. We use Adam (Kingma and Ba, 2014) optimizer for optimization.

3.2 Model 2

We propose another approach whose embedding layer is based on *Embedding for Language Model (ELMo)* (Peters et al., 2018) and the MLP Network, which is same as we applied in Model 1. The diagram of this model is shown in the Figure 3.

Embedding Layer: Embedding from Language Model (ELMo) has several advantages over the other word vector methods, and found to be a good performer in many challenging NLP problems. It has key features like i. Contextual i.e. representation of each word is based on entire corpus in which it is used ii. Deep i.e. it combines all layers of a deep pre-trained neural network and iii. Character based i.e. it provides representations which are based on character, thus allowing the network to make use of morphological clues to form robust representation of out-of-vocabulary tokens during training. The ELMO embedding is very efficient in capturing context. The multi-domain datasets are having different vocabularies and contexts, so we make use of such a word vector representation method to capture the context. News topics and corresponding documents are given to Elmo Embedding model. This embedding layer produces the representation for news topic and news content.

After getting the embedding of the topic and the context, we merge them. The merged vector is fed into a five layers MLP (same as the previous model). Finally, we classify with a final layer having softmax activation function.

Dataset	# of Examples	Avg.words/sent	Words	Label
FakeNewsAMT	240	132/5	31,990	Fake
	240	139/5	33,378	Legit
Celebrity	250	399/17	39,440	Fake
	250	700/33	70,975	Legit

Table 1: Class Distribution and Word Statistics for Fake News AMT and Celebrity Datasets. Avg: Average, sent: Sentence

4 Experiments, Results and Discussion and Comparison with State-of-the-Art

Overall we perform four sets of experiments. In the following sub-sections we describe and analyze them one by one after the description of the datasets used.

Data: Prior datasets and focus of research for fake information detection are on political domain. As our research focus is on multiple domains, we foster the dataset released by Pérez-Rosas et al. (2018). They released two novel datasets, namely *FakeNews AMT* and *Celebrity*. The Fake News AMT is collected via crowdsourcing (Amazon Mechanical Turk (AMT)) which covers news of six domains (i.e. Technology, Education, Business, Sports, Politics, and Entertainment). The Celebrity dataset is crawled directly from the web of celebrity gossips. It covers celebrity news. The AMT manually generated fake version of a news based on the real news. We extract the data domain wise to get the statistics of the dataset. It is observed that each domain contains equal number of instances (i.e. 80). The class distribution among each domain is also evenly distributed. The statistics of these two datasets is shown in the following Table 1.

The news of the Fake News AMT dataset was obtained from a variety of mainstream news websites predominantly in the United States such as the ABCNews, CNN, USAToday, New York

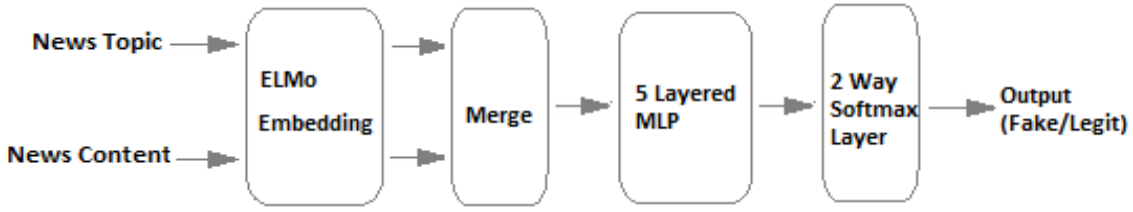


Figure 3: Architectural Diagram of the Proposed Second Model

Dataset	System	Model	Test Accuracy(%)
FakeNews AMT	Proposed	Model1	77.08
		Model2	83.3
	(Pérez-Rosas et al., 2018)	Linear SVM	74
Celebrity	Proposed	Model1	76.53
		Model2	79
	(Pérez-Rosas et al., 2018)	Linear SVM	76

Table 2: Classification Results for the FakeNews AMT and Celebrity News Dataset with Two Proposed Methods and Comparison with Previous Results

Training	Testing	Accuracy(%)
FakeNewsAMT	Celebrity	54.3
Celebrity	FakeNewsAMT	68.5

Table 3: Results Obtained in Cross-Domain Analysis Experiments on the Best Performing System.

Times, FoxNews, Bloomberg, and CNET among others.

Multi-Domain Analysis: In this section, we do experiments on whole Fake News AMT and Celebrity datasets individually. We train our models on the whole Fake News AMT and Celebrity dataset and test on the respective test set. As the datasets is evenly distributed between real and fake news item, a random baseline of 50% could be assumed as reference. The results obtained by the two proposed methods outperform the baseline and the results of Pérez-Rosas et al. (2018). The results obtained and comparisons are shown in the Table 2. Our results indicate this task could be efficiently handled using deep learning approach.

Cross-Domain Analyses: We perform another set of experiment to study the usefulness of the best performing system (i.e. Model2) across the domains. We train the model2, on FakeNews AMT and test on Celebrity and vice-versa. The results are shown in the Table 3. If we compare with the in domain results it is observed that there is a significant drop. This drop also observed in the work of Pérez-Rosas et al. (2018) in machine learning setting. This indicates there is a significant role of a domain in fake news detection, as it is established by our deep learning guided experiments too.

Multi-Domain Training and Domain-wise Testing: There are very small number of examples pairs in each sub-domain (i.e. Business, Technology etc) in FakeNews AMT dataset. We combine the examples pairs of multiple domains/genres for cross corpus utilization. We train our proposed models on the combined dataset of five out of six available domains and test on the remaining one. This has been performed to see how the model which is trained on heterogeneous data react on the domain to which the model was not exposed at the time of training. The results are shown in *Exp. a* part of the Table 4. Both the models yield the best accuracy in the *Education* domain, which indicates this domain is open i.e. linguistics properties, vocabularies of this domain are quite similar to other domains. The models (i.e. Model 1 and 2) perform worst in the *Entertainment* and the *Sports*, respectively, which indicate these two domains are diverse in nature from the others in terms of linguistics properties, writing style, vocabularies etc.

Domain-wise Training and Domain-wise Testing: We also eager to see in-domain effect of our systems. The FakeNews AMT dataset comprises of six separate domains. We train and test our models, on each domain's dataset of Fake News AMT. This evaluates our model's performance domain-wise. The results of this experiment are shown in the *Exp. b* part of the Table 4. In this case both the models produce the highest accuracy in the *Sports* domain, followed by the *Entertainment*, as we have shown in our previous experiments that these two domains are diverse in nature from the others. This fact is established by this experiment too. Both the models produce the lowest result in the *Technology* and the *Business* domain, respectively.

Visualization of Word Level Attention: We take the visualization of the topic and the corresponding document at word level attention as shown in the Figure 4 and 5, respectively. The aim is to visualize the words which are assigned more weights during the prediction of the output class.

Domain	Exp. a		Exp. b	
	Model1	Model2	Model1	Model2
Business	74.75	78.75	63.56	68.56
Education	77.25	91.25	65.65	70.65
Technology	76.22	88.75	64.3	65.35
Politics	73.75	88.75	64.27	69.22
Entertainment	68.25	76.25	65.89	71.2
Sports	70.75	73.75	67.86	71.45

Table 4: Result of Exp. a (Trained on Multi-domain Data and Tested on Domain wise Data) and Exp. b (Trained on Domain wise Data and Tested on Domain wise Data)

Figure 4: Word Level Attention on News Topic

In these Figures, words with more deeper colour indicate that they are getting more attention. We can observe, the words *secretary*, *education* in 4 and *President*, *Donald* in 5 are the words having deeper colour, i.e. these words are getting more weight compared to others. These words are Named Entities (NEs). It could be concluded that NEs phrases are important in fake news detection in multi domain setting.

4.1 Error Analysis

We extract the mis-classified and also the truly classified instances produced by the best performing system. We perform a rigorous analysis of these instances and try to find out the pattern in the mis-classified instances and the linguistics differences between those two categories of instances. It is found that the model fails mostly in the *Entertainment* followed by the *sports* and the *Business* domain etc. To name a few, we are showing such examples which are actually "Legitimate", but predicted as "Fake" the Table 5 and which are actually "Fake", but predicted as "Legitimate" the Table 6. It is observed that both the topic and document are having ample number of NEs. It needs further investigation in this font.

5 Conclusion and Future work

In this article, we propose two deep learning based approaches to mitigate the problem of fake news detection from multiple domains platform. Antecedent works in this line pay attention on satirical news or made use of the content of the fact-checking websites, which was restricted to one domain (i.e. politics). To address these limitations, we focus to extend this problem into multi domain scenario. Our work extends the concept of fake news detection from uni-domain to multi-domain, thus making it more general and realistic. We evaluate our proposed models on the datasets whose contents are from multiple domains. Our two proposed approaches outperform the existing models with a notable margin. Experiments also reveal that there is a vital role of a domain in context of fake news detection. We would like to do more deeper analysis of the role of domain for this problem in future. Apart from this our future line of research would be as follows:

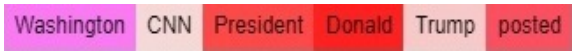


Figure 5: Word level Attention on News Document, A Part of it is Shown Due to Space Constraint.

- It would be interesting for this work to encode the domain information in the Deep Neural Nets.
- BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019) embedding based model and make a comparison with fastText and ELMo based models in the context of fake news detection.
- Use of transfer learning and injection external knowledge for better understanding.
- Handling of Named Entities efficiently and incorporate their embedding with the normal phrases.
- Using WordNet to retrieve connections between words on the basis of semantics in the news corpora (both topic and document of news) which may influence in detection of Fake News.

6 Acknowledgement

This work is supported by **Elsevier Centre of Excellence for Natural Language Processing** at Computer Science and Engineering Department, Indian Institute of Technology Patna. Mr. Tanik Saikh acknowledges for the same.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. *International Conference on Learning Representation (ICLR)*.

Domain	Topic	Content
Entertainment	Chris Pratt responds to body shamers telling him he's too thin	Big or small Chris Pratt has heard it all. These days the ”Guardians of the Galaxy” star 37 is taking flak for being too thin but he's not taking it lying down. Pratt who has been documenting the healthy snacks he's eating while filming ”Jurassic World 2” in a series of ”What's My Snack” Instagram videos fired back – in his usual tongue-in-cheek manner – after some followers apparently suggested he looked too thin. ”So many people have said I look too thin in my recent episodes of #WHATSMYSNACK he wrote on Instagram Thursday. Some have gone as far as to say I look 'skeletal.' Well just because I am a male doesn't mean I'm impervious to your whispers. Body shaming hurts.”
Business	Banks and Tech Firms Battle Over Something Akin to Gold: Your Data	The big banks and Silicon Valley are waging an escalating battle over your personal financial data: your dinner bill last night your monthly mortgage payment the interest rates you pay. Technology companies like Mint and Betterment have been eager to slurp up this data mainly by building services that let people link all their various bank-account and credit-card information. The selling point is to make budgeting and bookkeeping easier. But the data is also being used to offer new kinds of loans and investment products. Now banks have decided they aren't letting the data go without a fight. In recent weeks several large banks have been pushing to restrict the sharing of this kind of data with technology companies according to the tech firms. In some cases they are refusing to pass along information like the fees and interest rates they charge. Both sides see big money to be made from the reams of highly personal information created by financial transactions.

Table 5: Examples of mis-classified instances from Entertainment and Business domain. Examples are originally ”Legitimate” but predicted as ”Fake”.

Domain	Topic	Content
Sports	Slaven Bilic still has no support of West Ham's owners	”West Ham's owners have no faith in manager Slaven Bilic as his team won only six of their 11 games this year according to Sky sources. Bilic's contract runs out in the summer of 2018 and results have made it likely that he will not be offered a new deal this summer. Co-chairman David Sullivan told supporters 10 days ago after West Ham lost 3-2 at home to Leicester City. Sullivan said that even if performances and results improved in the next three games against Hull City Arsenal and Swansea City. West Ham's owners have a track record of being unloyal to their managers who don't meet their specs and there is a acceptance at boardroom level that Bilic has failed to prove a solid season.”
Education	STEM Students Create Winning Invention	STREAMWOOD, Ill. (AP) – A group of Streamwood High School students have created an invention that is exciting homeowners everywhere - and worrying electricity companies at the same time. The kids competed in the Samsung Solve for Tomorrow contest, entering and winning with a new solar panel that costs about $100 but can power an entire home - no roof takeover needed! The contest won the state-level competition which encourages teachers and students to solve real-world issues using science and math skills; the 16 studens will now compete in a national competition and, if successful, could win a prize of up to $200,000.

Table 6: Examples of mis-classified instances from the Sports and Education domain. Examples are originally ”Fake” but predicted as ”Legitimate” .

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Yimin Chen, Niall J Conroy, and Victoria L Rubin. 2015. News in an Online World: The Need for an automatic crap detector. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Niall J Conroy, Victoria L Rubin, and Yimin Chen. 2015. Automatic Deception Detection: Methods for Finding Fake News. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kaibo Duan, S. Sathiya Keerthi, Wei Chu, Shirish Krishnaj Shevade, and Aun Neow Poo. 2003. Multi-Category Classification by Soft-max Combination of Binary Classifiers. In *Proceedings of the 4th International Conference on Multiple Classifier Systems*, MCS'03, pages 125–134, Berlin, Heidelberg. Springer-Verlag.

William Ferreira and Andreas Vlachos. 2016. Emergent: a Novel Data-set for Stance Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1168, San Diego, California. Association for Computational Linguistics.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M. Meyer, and Iryna Gurevych. 2018. A Retrospective Analysis of the Fake News Challenge Stance-Detection Task. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1859–1874, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Gottfried Jeffrey and Shearer Elisa. 2016. News use Across Social Media Platforms 2016. In *In Pew Research Center Reports.*

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2018. Combining Fact Extraction and Verification with Neural Semantic Matching Networks. *arXiv preprint arXiv:1811.07039*.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6859–6866.

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. Association for Computational Linguistics.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic Detection of Fake News. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2016. Credibility Assessment of Textual Claims on the Web. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 2173–2178.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume*

1: Long Papers), pages 231–240, Melbourne, Australia. Association for Computational Linguistics.

Victoria Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake News or Truth? using Satirical Cues to Detect Potentially Misleading News. In *Proceedings of the second workshop on computational approaches to deception detection*, pages 7–17.

Victoria L. Rubin, Yimin Chen, and Niall J. Conroy. 2015. Deception Detection for News: Three Types of Fakes. In *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, ASIST '15, pages 83:1–83:4, Silver Springs, MD, USA. American Society for Information Science.

Tanik Saikh, Amit Anand, Asif Ekbal, and Pushpak Bhattacharyya. 2019. A Novel Approach Towards Fake News Detection: Deep Learning Augmented with Textual Entailment Features. In *International Conference on Applications of Natural Language to Information Systems*, pages 345–358, Salford, UK. Springer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-Scale Dataset for Fact Extraction and Verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426, Vancouver, Canada. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural Image Caption Generation with Visual Attention. In *International conference on machine learning*, pages 2048–2057.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Wenpeng Yin and Dan Roth. 2018. TwoWingOS: A two-wing optimization strategy for evidential claim verification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 105–114, Brussels, Belgium. Association for Computational Linguistics.

Samajh-Boojh: A Reading Comprehension System in Hindi

Shalaka Vaidya[*1], **Hiranmai Sri Adibhatla**[*2], **Radhika Mamidi**[3]
Language Technologies Research Centre
Kohli Center on Intelligent Systems
International Institute of Information Technology - Hyderabad
[1]hello@shalakavaidya.me
[2]hiranmai.sri@research.iiit.ac.in
[3]radhika.mamidi@iiit.ac.in

Abstract

This paper presents a novel approach designed to answer questions on a reading comprehension passage. It is an end-to-end system which first focuses on comprehending the given passage wherein it converts unstructured passage into a structured data and later proceeds to answer the questions related to the passage using solely the aforementioned structured data. To the best of our knowledge, the proposed model is first of its kind which accounts for entire process of comprehending the passage and then answering the questions associated with the passage. The comprehension stage converts the passage into a **Discourse Collection** that comprises of the relation shared amongst logical sentences in given passage along with the key characteristics of each sentence. This model has its applications in academic domain and query comprehension in speech systems among others.

1 Introduction

The Samajh-Boojh[1] system which we have built focuses on the basic principles behind utilization of rules in order to capture the semantics of the given passage which is in the Devanagari script. The current trend is towards incorporating machine learning in the question answering models but they come with a downside of requiring huge quantity and variety of training data to achieve decent accuracy. Whereas the proposed model is rule-based and hence eliminates the need for extensive training data while still providing 75% accuracy.

The Samajh-Boojh system answers 11 types of questions (Table 1) using approximately 25 rules. This sheds light on the fact that, with substantially less number of rules a wide range of questions can be answered. It is an extension to Prashnottar model (Sahu et al., 2012) which could handle 4 types of questions using 4 rules. The system can be classified into two parts, the comprehension part and the question answering part, these two parts together ensure that the system behaves similar to the way humans approach the questions which are asked based on reading comprehension passage. The comprehension part of the system converts the given passage whose inherent structure cannot be grasped by the machine to a structured and machine extractable data called Discourse Collection. The Discourse Collection is then sent to the QA system as an input along with the query to obtain the relevant answer. This feature sets the proposed model apart from the commonly used information retrieval and extraction based techniques.

The Panchatantra collection[2] comprising of 65 short stories was used to experiment on the model. This dataset had variety of stories with different lengths. The questions on each of these stories were framed by multiple annotators and best of which were picked to validate the system. The answers given by the annotators were used as gold data to measure the quality of the system.

2 Architecture and Design

The Samajh-Boojh System is broadly classified into two parts: comprehension part and question answering part. The system works similar to human approach of answering reading comprehension questions. The system

[*]equal contribution

[1]Samajh-Understanding and Boojh-Analysis, which translates to reading comprehension in Hindi.

[2]https://www.hindisahityadarpan.in/2016/06/panchatantra-complete-stories-hindi.html

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 239–248
Hyderabad, India, December 2019.

takes passage and queries corresponding to the passage as the input, and returns answers to the questions. In subsequent sections we dwell deeper into these parts and see how they function.

2.1 Reading Comprehension Part

The reading comprehension part of the system is responsible for structuring the story into a **Discourse Collection** which contains the characteristics of the story. This structure is inspired by Thorndyke's Story Grammar (Thorndyke, 1977). Discourse Collection is comprised of the following components:

Episode_Id: Unique key associated to the sentence. Its incrementally assigned as we parse the sentences. In Discourse Collection, we associate each logical sentence as an episode.

Original_sentence: The WX version of the logical sentence found in the story.

Time: The time setting in which the episode took place. If the Original_Sentence doesn't specify the time, this field is populated from the previous episode's Time value. Default is 'tbd: to be decided'

Location: The place in which the episode took place. If the Original_Sentence doesn't specify the location, this field is populated from the previous episode's Location value. Default is 'tbd: to be decided'

Karta: The karta (doer) of the logical sentence is given as the value. This is obtained from the dependency parser[3].

Karta_Adpos: The Adpos (adjective and prepositions) associated with the Karta to frame the answers during the question-answering stage.

Karma: The karma of the logical sentence is populated in this column.

Karma_Adpos: The Adpos (adjective and prepositions) associated with the Karma to frame the answers during the question-answering stage.

Anaphora_Resolved_Sentences: The logical sentence in which the anaphora is replaced with noun.

Root_Node_Sentences: The words of the logical sentences are replaced with their roots. For this we used the shallow parser [4].

Given: The sentence which is related to the current sentence.

New: The current sentence which is having the Given sentence as a prerequisite.

Parser_Output: The output of the dependency parser.

The overview of the Reading Comprehension system is seen in Figure 1. The passage is given as the input to **Logical Sentence Module** to break it into logical sentences, the split passage is given as input to the Discourse Generator module which contains **Graph Maker Module**, **Anaphora Resolution Module**, **Root Node Resolution Module** and **Discourse Information Filler Module**. The final output of these four modules is the Discourse Collection which is the output of comprehension system. The individual modules of the reading comprehension system along with detailed working is explained in the forthcoming sections.

Logical Sentences Module

Even though the passage can easily be split into words, sentences and paragraphs when given as the input, it's a challenge to extract the semantics. We break the sentences based on the generic punctuation marks such as full stop, comma, semicolon, question mark, exclamation mark and conjunctions such as और, कि, पर, कर, फिर, इसीलिए, तब, तो, क्योंकि, क्यूंकि, लेकिन, परंतु, किन्तु.

When splitting the sentences by the split words, we noticed the tags were improper at some instances. Example:
S1:राम घर जाना चाहता था पर नहीं जा पाया।
T1: Ram wanted to go home but couldn't go.
S2:राम घोड़े पर बैठा था।
T2: Ram sat on the horse.

Here the word पर translates to 'but' and 'on', we want to split the sentence into two parts only in S1 and not in S2. The POS tags using Dependency Parser[3] give 'PSP' tag, hence making it difficult to differentiate on which पर to split the sentences. To resolve this issue we decided to see the context of the given split word and then make the decision. The logic for deciding whether to split or not is given

[3]https://bitbucket.org/iscnlp/parser/src/master/README.rst

[4]http://ltrc.iiit.ac.in/analyzer/hindi/index.cgi

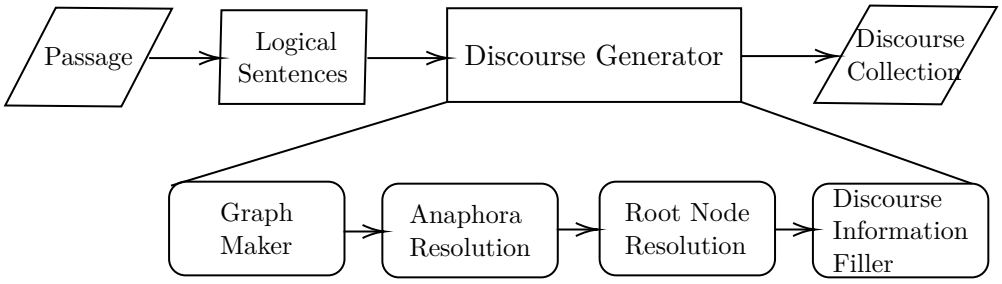


Figure 1: Comprehension Design

below:
If the word before the split word is verb ie. having POS tag as 'VM' or 'VAUX' then we split the sentence into two parts and populate the Discourse Collection with two episodes each containing the split sentences as Original_Sentences(OS).
In the Figure 2, we see in the sample passage that apart from the full stops, the sentences are split at 'taba/तब' and 'para/पर' resulting in 5 episodes.

Graph Maker

The Original_Sentence which was populated in Logical Sentence Module is sent through the dependency parser [3] and the parser output is stored in the Discourse Collection of the corresponding episode. From the parser output, if there exits any **k7t** relation, it is stored in the 'Time' slot of the episode. If there exists a **k7p** relation, it is stored in the 'Location' slot of the episode. The word with **k1** relation is stored in the 'Karta' slot of the episode along with the 'lwg___psp' as case marker of the Karta and word with relation 'nmod___adj' as the Karta adjective, the case marker and adjective with Karta word are called Karta_adpos and stored in the corresponding episode. The word with **k2** relation is stored in Karma slot of the episode. The child nodes of the Karma in dependency tree who have the relations 'lwg___psp' and 'nmod___adj' are stored as Karma_Adpos.

Anaphora Resolution Module

We use the Original_Sentence from the episode to resolve the anaphora and store it in Anaphora_Resolved_Sentence(ARS) of the corresponding episode. We used the algorithm mentioned in Dakwale(2014) to resolve the anaphora. This algorithm is a right fit as it uses rules from the dependency parser[3].
We see in Figure 2 that the the word 'vaha/वह' translates into rAma in episode 2 and billI in episode 4 based on the context, 'usE/उसे' and 'vO/वो' are resolved into rAma in episode 2 and 5.

Root Node Resolution Module

The root of a word is important when we are comparing two sentences. We convert each word in Anaphora_Resolved_Sentence of the episode into its root form and store it as Root_Node_Sentence(RNS) for corresponding episode. We used the IIIT Parser[4] and the output was parsed through the SSF format mentioned in Bharati(2007) and the root form of the words were extracted.
In episode 2 the word 'gaya' changes to 'ja' in Figure 2.

Discourse Information Filler

Discourse is when we look beyond the scope of a sentence and use information between their relation. Here we fill the 'Given' and 'New' values of the episode. The default values are 'tbd: to be decided'. If the passage sentence has split words mentioned in section 2.1, the sentence is split into two episodes such that, the second episode will contain the first split sentence as 'Given' in its Discourse Collection and second split sentence as the 'New'. Our assumption and complexity is limited to identifying a co-dependency between two sentences if they are separated by split words.
The output of this stage will give the Discourse Collection. Episode 2 in the Figure 2, has 'Given' and 'New' values populated since it has the word 'taba/तब' (translation: then).

Passage:

rAma Eka acchA ladkA thA. vaha Eka dina pAThaSAlA jA rahA thA taba usE Eka billI dikhI. rAma usakE pAsa gayA para vaha bhAga gaI aura vO dukhI hO gayA

Discourse Collection:


```
{
"0": {
      "OS": "rAma Eka acchA ladkA thA",
      "karta": "rAma",
      "kartaadj": ["rAma", "acchA"],
      "ARS": "rAma Eka acchA ladkA thA",
      "RNS": "rAma Eka acchA ladka thA"
       }
"1": {
      "OS": " vaha Eka dina pAThaSAlA jA
              rahA thA",
      "time": "din",
      "location": "pAThaSAlA",
      "karta": "rAma",
      "ARS": "rAma Eka dina pAThaSAlA jA
              rahA thA",
       "RNS": "rAma Eka dina pAThaSAlA jA
              rahA thA"
       }
"2": {
      "OS": "usE Eka billI dikhI",
      "time": "din",
      "location": " pAThaSAlA",
      "karta": "billI",
      "given": " rAma Eka dina pAThaSAlA jA
                rahA thA",
      "new": "rAma Eka billI dikhI",
      "ARS": "rAma Eka billI dikhI",
      "RNS": "rAma Eka billI dikha"
       }
"3": {
      "OS": " rAma billI pAsa gayA",
      "time": "din",
      "location": " billI pAsa",
      "karta":"rAma",
      "ARS": "rAma billI pAsa gayA",
      "RNS": "rAma billI pAsa jA"
       }
"4": {
      "OS": "vaha bhAga gaI",
      "time": "din",
      "location": " billI pAsa",
      "karta": "rAma",
      "given": " rAma billI pAsa gayA",
      "new": "billI bhAga gaI",
      "ARS": "billI bhAga gaI",
      "RNS": "billI bhAga jA"
       }
"5": {
      "OS": "vO dukhI hO gayA",
      "time": "din",
      "location": " billI pAsa",
      "karta": "rAma",
      "given": "billI bhAga gaI",
      "new": "rAma dukhI hO gayA",
      "ARS": "rAma dukhI hO gayA",
      "RNS": "rAma dukhI hO jA"
       }
}
```


Figure 2: Discourse Collection
The passage and its corresponding discourse collection is shown here. Only the populated values are shown, rest all are 'tbd-to be decided' except for parser_output. OS-original_sentence, ARS-Anaphora_Resolved_Sentence, RNS-Root_Node_Sentence

Question Type	Question Words
Karta	'kisnE' 'kisakE' 'kauna' 'kisasE'
Karma	'kisakO' 'kisakI'
Time	'kaba' 'samaya' 'dina'
Loc	'kahA '
Recipient	'kisE'
Adj_Noun	'kaisA' 'kaisI'
Intf	'kitnA' 'kitnE'
Kya	'kyA'
Kiske	'kiskE'
Kiska	'kiskA' 'kiskI'
GivenNew	'kara' 'bAda' 'phalE' 'kyoM'

Table 1: Types of Questions.

2.2 Question Answering Part

The Question-Answering part of this model takes Discourse Collection which was output of the Reading Comprehension part, as the input along with the query related to the passage and returns the answer as the output. The brief overview of the system is shown in Figure 3. The query is fed in the Devanagari format and the answer is given in the same. The working of this system is seen in following sections.

Question Analyzer Module

This module takes the Devanagari format of the input query and returns the type of the question along with key word relevant to the query. This format is similar to that found in QLL (Vargas-Vera et al., 2003).

We tag the questions based on the question words into 11 major categories shown in Table 1. This list can be expanded as per the required answers from the question word. For example,
Q1: गांव में कितने मुर्गे रहते थे?
T1: How many chickens were there in the village?
The answer expects the quantity of the chickens. So, we place it into 'Intf' category

We now see each type of question in detail and see how they are handled:
Karta: It involves the question words which requires the answer as the doer. किसने बंदर को

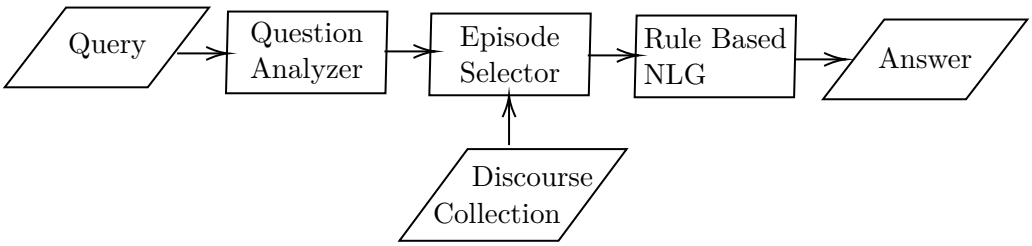


Figure 3: Question Answering Design

परेशान किया? (Who troubled the monkey?)
Karma: The question which requires the answer as the act of the sentence. बल्ली किसको दिखी? (Who saw the cat?)
Time: The question which expects the answer as the time. सूरज कब घर आया? (When did Suraj come home?)
Loc: The question word which requires answers as the location. किताब कहाँ राखी थी? (Where was the book kept?)
Recipient: The question which expects the answer of the receiver.किसे चोट लगी थी? (Who got injured?)
Adj_Noun: The question which requires the answer as the adjective of particular noun.राम कैसा लड़का था? (What kind of boy was Ram?)
Intf: Question which requires the answer as the quantity of a particular entity. एक गांव में कितने लोग थे? (How many people stayed in the village?)
Kya: When the answer is supposed to describe or explain the situation. श्याम ने खाने में क्या खाया? (What did Shyam have for his meal?)
Kiske: When the question requires the entities involved with mentioned subject as the answer. सीता किसके साथ खेल रही थी? (With whom was Sita playing?)
Kiska: The question is seeking the possessive trait of the entity mentioned. यह किताब किसकी है? (Whose book is this?)
GivenNew: The question that gives an activity and requests for the *consequence* of the activity falls in category of GivenKnown. The questions which describe an activity and expects the *cause* of the activity as the output, it falls in category of NewKnown.
Example:
कुछ लड़कों ने एक बिल्ली को तंग किया और वह परेशान हो गयी (Some boys troubled a cat and the cat got angry)
Given Known:
बिल्ली तंग होने के बाद क्या हुआ? (What Happened after the cat was troubled?), Given Info from the question: बिल्ली तंग हुई(Cat was troubled) Here, the answer is expected to be the consequences of cat being troubled.
New Known:
बिल्ली परेशान क्यों हो गयी? (Why was the cat angry?), new info from the question: बिल्ली परेशान हुई (cat is angry) Here, the answer is expected to address the reasons why the cat was angry. The output of the question analyzer module for above mentioned question types is shown in Figure 2.

Question Type	**Output Format**
Karta	[‘Karta’]
Karma	[‘Karma’]
Time	[‘Time’]
Loc	[‘Loc’]
Recipient	[‘Recipient’]
Adj_Noun	[‘Adj_Noun’, one word before the question word]
Intf	[‘Intf’, one word after the question word]
Kya	[‘Kya’]
Kiske	[‘Kiske’, one word after the question word]
Kiska	[‘Kiska’]
GivenNew	[‘GivenNew’, the information which is either new or given]

Table 2: Output of the Question Analyzer.

There is a preference order given to these question types, in cases of when two words belonging to two different classes (in Table 1) exist in same query. The observed overlaps in-

clude: Time and Kya: in this case the question type will be treated as Kya. Kya and Given-New: in case of this overlap, the question type will be treated as Kya.

Episode Selector Module

The input to this module is the output of the Query Analyzer module (Table 2) and Discourse Collection(Figure 2) which is the output of the Reading Comprehension stage of our system. This is seen clearly in Figure 3. The episode is detected by using Jaccard similarity between the Query(Q) and Root_Node_Sentence(RNS) whose formula is as follows:

$$Jaccard\_Sim(Q, RNS) = \frac{n(Q \cap RNS)}{n(Q \cup RNS)}$$

Where, $n(Q \cap RNS)$ is number of common words in the Query and Root_Node_Sentence and $n(Q \cup RNS)$ is the total number of words in Query and Root_Node_Sentence.

Algorithm 1 Weighted Jaccard Similarity

```
procedure JaccardSim(Q, RNS)
    NounTags ← {MNP, MNS, NN}
    VerbTags ← {VM}
    VAuxTags ← {VAUX}
    AdTags ← {JJ, RB}
    Union ← Q ∪ RNS
    Intersec ← Q ∩ RNS
    JS ← 0
    for all word ∈ Intersec do
        switch POS(word) do
            case ∈ AdTags
                JS ← JS + 5
            case ∈ VerbTagss
                JS ← JS + 4
            case ∈ NounTags
                JS ← JS + 3
            case ∈ VAuxTags
                JS ← JS + 2
            case default
                JS ← JS + 1
    end for
    NormalisedJS ← JS/|Union|
    return JS, NormalisedJS
end procedure
```

We have modified the formula to give better results. The formula is our version of weighted Jaccard similarity, wherein, we take the POS tags of the words which are common between the Query and the Root_Node_Sentence, and give more weightage to the word if it is less frequent, rather than focusing on frequently occurring words, which don't capture the similarity between the Query and Root_Node_Sentence such as prepositions. We have given the priority to rare words based on their POS word tags. Priorities of POS tags as given as Adjective/Adverb > Verb > Noun > Auxillary Verb > Others.
The respective POS tags from the parser are: $(JJ/RB) > (VM) > (NN/NNS/NNP) > (VAUX) > Others$. We call this as the Jaccard_Score between the episode and the Query, if we divide Jaccard_Score by $(Q \cap RNS)$, we get Normalized_Jaccard_Score.
After calculating the Jaccard_Score and the Normalized_Jaccard_Score for each episode in Discourse Collection, we take the episode which has the highest Jaccard_Score, if two episodes have highest Jaccard_Score, we compare their Normalized_Jaccard_Score and choose the higher valued episode as our chosen episode. The pseudo code is given in Algorithm 1.
The Output of this module is the Episode_Id(from the Discourse Collection) which has maximum similarity to the Query.

Rule Based Natural Language Generator Module

This Module generates answer for a given query. It takes the episode chosen by the Episode Selector, Discourse Collection, and the query as input and generates the answer according to the query type.
Answer for various question types(Table 2) is generated as follows:
Karta: We extract the Karta from the Discourse_Collection for the chosen episode along with Karta_Adpos(Karta_Adjective and Karta_Case_Markers) and give the answer as Karta_Adjective + Karta + Karta_Case_Marker. The answer to the question kIsnE zyAma kO kalama dI? [Karta Question Type] answer will be rAma nE (refer Figure 4).
Karma: We extract the Karma from the

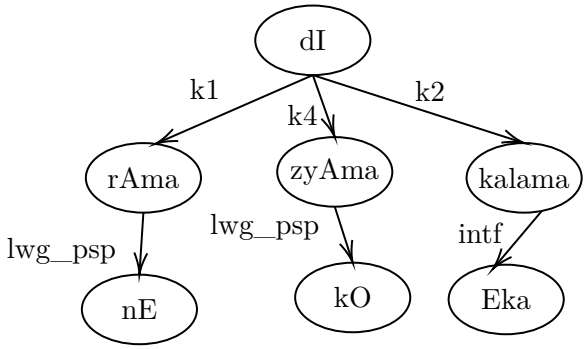


Figure 4: Karta, Recipient, Intf and Kya

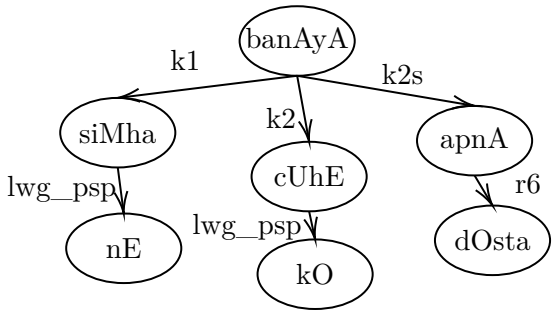


Figure 5: Karma

Discourse_Collection for the chosen episode along with Karma_Adpos(Karma_Adjective and Karma_Case_Markers) and give the answer as Karma_Adjective + Karma + Karma_Case_Marker. If the 'Karma' slot of the Discourse_Collection isn't populated, the Anaphora_Resolved_Sentence of the chosen episode is returned. The answer to the question siMha ne kiskO apnA dosta banAyA? [Karma Question Type] answer will be cUhE kO (refer Figure 5).

Time: If the Parser_Output of the given episode contains the 'k7t' relation between two nodes, then the child node is the output. If there doesn't exist any 'k7t' relation, then 'k7' relation is used and the child node is the answer. If either of these aren't existing, then time is extracted from the 'Time' slot of the Discourse_Collection of the given episode and is displayed as the answer. The answer to the question sIta kaba pathzAlA gayI? [Time Question Type] answer will be subah (refer Figure 6).

Loc: If the Parser_Output of the given episode contains the 'k7p' relation between two nodes, then the child node is the output. If there doesn't exist any 'k7p' relation, then 'k7' relation is used and the child node is the

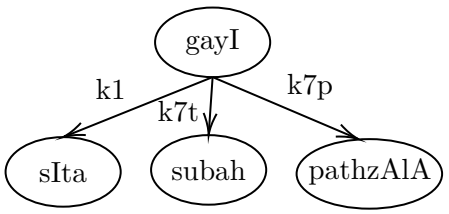


Figure 6: Loc and Time

answer. If either of these aren't existing, then location is extracted from the 'Loc' slot of the Discourse_Collection of the given episode and is the answer. The answer to the question sIta kaha gayI? [Loc Question Type] answer will be payhzAla (refer Figure 6).

Recipient: The main verb(MV) is extracted from the Parser_Output of the Episode. If the MV shares relation 'k4' with a child we return that child as the answer. If there doesn't exist any child with 'K4' relation, we check for any child nodes of the MV with 'k4a' relation and return that as the answer. The answer to the question rAma ne kisE kalama dI? [Recipient Question Type] answer will be zyAma (refer Figure 4).

Adj_Noun: From Table 2, we can see that the output is the Noun whose Adjective is asked in the question. Let the Noun whose adjective is asked be MN. We take the Parser_Output of the chosen episode and check for the child nodes of the MN who have relation 'nmod__adj' and return the child node as the answer. The answer to the question kalama kaisI hai? [Adj_Noun Question Type] answer will be suNdara (refer Figure 8).

Intf: From the 2, we can see that the Noun(MN) whose quantity is asked is returned along with the classification of the question. To get the answer we take the Parser_Output of the chosen episode and search for the MN, then we check for child nodes of MN who have relation 'intf' with MN, lets call the child 'NounIntf', we then check child nodes who have relation 'nmod__adj' with MN, lets call it 'NounAdj'. If 'NounIntf' and 'NounAdj' exist, we return the answer as NounIntf + NounAdj and MN as the answer. If 'NounAdj' doesn't exist, we just return NounIntf + MN as the answer. The answer to the question rAma ne kitnE kalama dIyE? [Intf Question Type] answer will be Eka (refer Figure 4).

Kya: We extract the Main Verb (MV) from

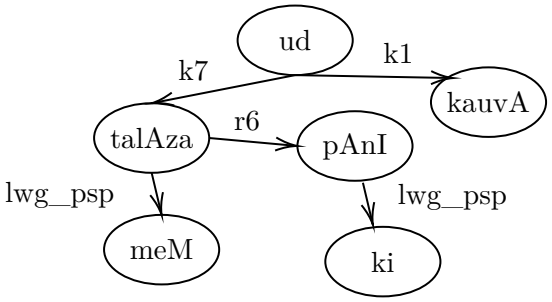


Figure 7: Kiske

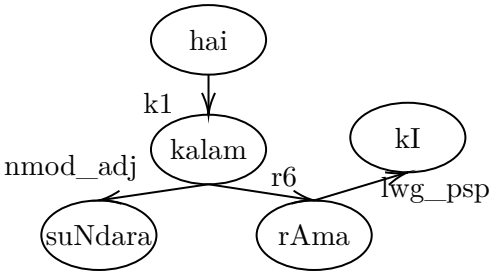


Figure 8: Kiska, Adj_Noun

the Parser_Output of the Discourse Collection of the chosen episode. We then check if either of 'k1s', 'pof', 'k2' relations exist between the MV and its children. If it exists, we check if the child is mentioned in the question, if it isn't mentioned in the question, we return child Node as the answer. If no child exists with above mentioned relations or if it exists and the child has occurred in the question itself then, we check if the 'Given' slot of the discourse Collection is populated for the next episode and return the value in 'NEW' slot of the Discourse Collection as the answer. The answer to the question rAmA nE kyA dIyA? [Kya Question Type] answer will be kalama (refer Figure 4).

Kiske: From Table 2, we observe that subject whose entity is asked is known. We consider this subject as Main Noun(MN), we check the children of MN who have relation 'k7' and return it as the answer. If this relation doesn't exist, we check for children with 'r6' relation and return it as the answer. 'r6' and 'k7' are called *SambandhRelations* in Panninian Grammar (Bharati et al., 1995). The answer to the question kauvA kiske talAza meM udA? [Kiske Question Type] answer will be pAnI (refer Figure 7).

Kiska: We extract the Main Verb(MV) from the Parser_Output of the Discourse Collection for the particular episode. We check the children of MV which have relations in order, 'k2', 'k7', 'r6'. The answer to the question yaha kalama kiskI hai? [Kiska Question Type] answer will be rAma (refer Figure 8).

GivenNew: We can see from Table 2, the information of GivenNew is given as the output of the Query Analyzer step. If the question is 'Given Known' (Refer 2.2), we choose the episode based on the highest Jaccard similarity (mentioned in 2.2) between the 'Given' slot of the Discourse Collection and the query. The Episode which gets the highest score, is the chosen episode and we return the 'New' slot value as the answer. Similarly for the 'New Known' (Refer 2.2) we choose the episode based on the 'New' slot and the answer is in 'Given' slot of the same episode. The answer to the question jaba rAma billI kE pAsa gayA taba kyA huA? [GivenKnown Question Type] the answer is billI bhAga gaI. (refer Figure 2)

Default: In case an unknown question type is encountered or no answer has been given for the above question types, we return the Anaphora_Resolved_Sentence of the chosen episode as the answer.

3 Experiments

Panchatantra is a collection of fables. It has five parts, Mitra-Bheda (The loss of friends), Mitra-laabha (The winning of friends), Kakolukiyam (on crows and owls), Labdhapranasam(Losing what you have gained) and Apariksitakarakam (Ill-Considered actions). We have chosen a corpus of 65 stories from the tales across all parts of Panchantantra to test our system.

We collected the stories from the link mentioned in footnote[2] and fixed the syntax (punctuation and spellings). We annotated 440 questions for the above stories and assigned the question types, the ideal episode to be selected from the Discourse Collection and the correct answer for each of the questions. While annotating the questions, we made a conscious effort to involve more questions in type 'Kya' and 'GivenNew' (refer Table 4), since they are versatile concepts and we intended to test them extensively against the rules we formatted as the other types are more or less intuitive on the dependency parser tags[3]. We randomly

Story	Original Sentence	Anaphora Resolved Sentence	Root Node Sentence
1	3/7	5/7	5/7
2	7/11	7/11	11/11
3	3/6	3/6	4/6
4	3/5	4/5	4/5
5	1/8	3/8	4/8
6	1/3	2/3	2/3

Table 3: Episode selection accuracy

selected 10 stories from the corpus along with the questions and generated rules based on linguistic heuristics, to avoid overfitting. We then verified the rules on the remaining stories.

Out of 440 questions, 72 more questions were rightly answered on using weighted Jaccard similarity when compared to normal Jaccard similarity. That is, *16% episodes were rightly selected when weighted Jaccard similarity was used instead of normal Jaccard similarity on our model.*

We also compared episode selection accuracy by including modules mentioned in figure 1 on 6 random stories which are different from the previously selected stories. The results can be seen in table 3 for the same. It can be observed that the weighted Jaccard similarity accuracy improved consistently as we added the Root Node Resolution and then the Anaphora Resolution modules. The below example demonstrates the improvement in episode selection accuracy for different modules:

Question: राम क्या खा रहा था? [Translation: What was Ram eating?]

Original Sentence: उसने आम खाया. [Translation: He ate a mango] **Jaccard Score: 0**

Root Node Sentence: वह आम खा [Translation: He eat mango] **Jaccard Score: 4**

Anaphora Resolved Sentence: राम आम खा[Translation: Ram eat mango] **Jaccard Score: 7**

Overall accuracy of the answers based on question Types is Shown in Table 4. The figure clearly shows good accuracy for majority of the questions. Since the 'Kya' and 'GivenNew' format of the questions are versatile and the answers can be subjective, the accuracy for these categories cannot be comparable to the direct question types whose answers are obtainable through dependency parser tags solely. Overall accuracy of the system is *75.45%*.

Question Type	Questions	Accuracy
Karta	35	94.3%
Karma	7	100%
Time	7	100%
Loc	45	100%
Recipient	15	100%
Adj_Noun	15	100%
Intf	15	100%
Kya	179	71.6%
Kiske	13	84.62%
Kiska	5	100%
GivenNew	33	63.7%
Total	**440**	**75.45%**

Table 4: Accuracy of the answers

4 Future Work

This model currently doesn't answer कैसे [How] types of questions, which can be included in future.

Currently we don't resolve synonyms and antonyms to answer the questions, which when done, can improve Episode Selection algorithm and also aim at answering complex questions.

The current model assumes the passage to be in chronological order. We can improve the model if we capture the relative time of the episode to suite the passages which don't follow the chronological order.

Versatile questions such as 'GivenNew' and 'Kya' can be improved by increasing the scope of answer retrieval to multiple sentences or episodes around the selected episode unlike the single episode range implemented in our model. This will in-turn also increase the scope of model to include longer and complex texts.

5 Conclusion

Reading Comprehension is a complex task, which involves comprehending the passage and answering the questions following the passage. Once, we are able to structure this unstructured data(passage), we can answer the

questions relatively well without complex approaches. The rules in linguistic are intuitive and are capable of answering complex questions. Since it's rule based, there is no requirement of large data to obtain promising results. In the model we managed to get 75% accuracy with just 65 stories and managed to answer wide range of answers. This model is versatile and can be extended to other Indian languages provided the dependency parser (similar to one we used[3]) exists.

References

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

Akshar Bharati, Rajeev Sangal, and Dipti M Sharma. 2007. Ssf: Shakti standard format guide.

Praveen Dakwale. 2014. *Anaphora Resolution in Hindi*. Ph.D. thesis, PhD thesis, International Institute of Information Technology Hyderabad.

Shriya Sahu, Nandkishor Vasnik, and Devshri Roy. 2012. Prashnottar: a hindi question answering system. *International Journal of Computer Science & Information Technology*, 4(2):149.

Perry W. Thorndyke. 1977. Cognitive structures in comprehension and memory of narrative discourse. *Cognitive Psychology*, 9(1):77 – 110.

Dr. Maria Vargas-Vera, Enrico Motta, and John Domingue. 2003. Aqua: an ontology driven question answering system.

Author Index