Text Classification for Telugu: Datasets, Embeddings and Models for Downstream NLP Tasks

Thesis submitted in fulfillment
of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Engineering

by

Mounika Marreddy
20172152
mounika.marreddy@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
May 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "**Text Classification for Telugu: Datasets, Embeddings and Models for Downstream NLP Tasks**" by **Mounika Marreddy**, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Advisor: Prof. Radhika Mamidi

To My Family and Advisor

Acknowledgments

I thank my supervisor, Dr. Radhika Mamidi, for her continued genuine support throughout my PhD. I am thankful for her understanding of my situation and encouraging me on every step I achieved. She supported me and encouraged me during and after my pregnancy time which was so moving. I am grateful to her for all the interactive sessions that helped me identify my research plan, hurdles during the data preparation, discussing ideas, and drafting corrections. I am indebted to her for the way she gave me the freedom to explore ideas, listen to each one of them with patience, and provide helpful directions on how those ideas will work in the research direction. Thank you for all the research discussions and guidance for my further career plans.

I want to thank my mother, father, husband, and daughter, without whom I would not have completed this PhD journey. Thank you to each one of you for all your support and motivation to finish my PhD. I express my gratitude to my brother for his moral support during the initial days of my research. He gave me the confidence that I am capable of doing research.

I am fortunate to be a part of the Language Technologies Research Center, IIIT Hyderabad. This is where I got the confidence that I could learn new topics that help me do research. I am honored to interact with faculty like Prof. Dipti M Sharma, Dr. Vasudev Varma, and Dr. Manish Shrivatsava. Thank you, everyone, for sharing the knowledge to improve my academic and research skills. I want to thank research scholar Pruthwik for giving me all the necessary information throughout my PhD life.

I like to thank Subbareddy Oota, Lakshmi Vakada Sireesha, Venkata Charan Chinni, Suma Reddy, Radha Agarwal, and Aditya Avvaru for being my teammates. I am truly fortunate to have worked with the most flexible and hardworking people. I also want to thank my friends, lab mates, and my family members who supported and motivated me throughout the PhD journey at IIIT Hyderabad.

Abstract

Language understanding has become crucial in different text classification tasks in Natural Language Processing (NLP) applications to get the desired output. Over the past decade, machine learning and deep learning algorithms have been evolving with efficient feature representations to give better results. The applications of NLP are becoming potent, domain, and language-specific. For resource-rich languages like English, the NLP applications give desired results due to the availability of large corpora, different kinds of annotated datasets, efficient feature representations, and tools.

Due to the lack of large corpora and annotated datasets, many resource-poor Indian languages struggle to reap the benefits of deep feature representations. Moreover, adopting existing language models trained on large English corpora for Indian languages is often limited by data availability, rich morphological variation, syntax, and semantic differences. Most of the work being done in Indian languages is from a machine translation perspective. One solution is to use translation for re-creating datasets in low resource languages from English. But in case of Indian languages like Telugu, the meaning may change and some crucial information may be lost due to translation. This is because of their structural differences, morphological complexities, and semantic differences.

In this thesis, our main objective is to mitigate the low-resource problem for Telugu. Overall, to accelerate NLP research in Telugu, we present several contributions: (1) A large Telugu raw corpus of 80,15,588 sentences (16,37,408 sentences from Telugu Wikipedia and 63,78,180 sentences crawled from different Telugu websites). (2) Annotated datasets in Telugu for sentiment analysis, emotion identification, hate speech detection, sarcasm identification, and clickbait detection. (3) For the Telugu corpus, we are the first to generate pre-trained distributed word and sentence embeddings such as *Word2Vec-Te*, *GloVe-Te*, *FastText-Te*, *MetaEmbeddings-Te*, *Skip-Thought-Te*. (8) We pre-trained different contextual language models for Telugu such as *ELMo-Te*, *BERT-Te*, *RoBERTa-Te*, *ALBERT-Te*, *Electra-Te*, and *DistilBERT-Te*, word and sentence embeddings using graph-based models: *DeepWalk-Te* and *Node2Vec-Te*, and *Graph AutoEncoders (GAE)*. (4) We propose the multi-task learning model (MT-Text GCN) to reconstruct word-sentence graphs on TEL-NLP data while achieving multi-task text classification with learned graph embeddings. We show that our pre-trained embeddings are competitive or better than the existing multilingual pre-trained models: *mBERT*, *XLM-R*, and *IndicBERT*.

Lastly, the fine-tuning of pre-trained models show higher performance than linear probing results on five NLP tasks. We also experiment with our pre-trained models on other NLP tasks available in Telugu (Named Entity Recognition, Article Genre Classification, Sentiment Analysis, and Summarization) and find that our Telugu pre-trained language models (*BERT-Te* and *RoBERTa-Te*) outperform the state-of-the-art system except for the sentiment task. We hope that the availability of the created resources for different NLP tasks will accelerate Telugu NLP research which has the potential to impact more than 85 million people.

In this thesis, we aim at bridging the gap by creating resources for different NLP tasks in Telugu. These tasks can be extended to other Indian languages that are closer to Telugu culturally and linguistically by translating these resources without losing information like verb forms, cultural terms, vibhaktis etc. This is the first work that employs neural methods to Telugu language —a language that does not have good tools like NER, parsers and embeddings. Our work is the first attempt in this direction to provide good models in Telugu language by exploring different methods with available resources.

It can also help the Telugu NLP community evaluate advances over more diverse tasks and applications. We open-source our corpus, five different annotated datasets (SA, EI, HS, SAR, and clickbait), lexicons, pre-trained embeddings, and code here[1]. The pre-trained Transformer models for Telugu are available here[2].

[1]https://github.com/mounikamarreddy/NLP-for-Telugu-Language.git
[2]https://huggingface.co/ltrctelugu

Related publications

Journal Proceedings

1. **Mounika Marreddy**, Subba Reddy Oota, Lakshmi Vakada Sireesha, Venkata Charan Chinni, Radhika Mamidi,"Am I a Resource-Poor Language? Datasets, Embeddings, Models and Analysis for four different NLP tasks in Telugu Language," in ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)-2022.

Conference Proceedings

1. **Mounika Marreddy**, Subba Reddy Oota, Lakshmi Vakada Sireesha, Venkata Charan Chinni, Radhika Mamidi,"Multi-Task Text Classification using Graph Convolutional Networks for Large-Scale Low Resource Language," in International Joint Conference on Neural Networks (IJCNN)-2022.

2. **Mounika Marreddy**, Subba Reddy Oota, Lakshmi Vakada Sireesha, Venkata Charan Chinni, Radhika Mamidi,"Clickbait Detection in Telugu! Building from Scratch?," in International Joint Conference on Neural Networks (IJCNN)-2021.

3. Subba Reddy Oota, Aditya Avvaru, **Mounika Marreddy**, Radhika Mamidi, "Affect in Tweets using Experts Model," in Pacific Asia Conference on Language, Information and Computation (PACLIC)-2018.

Workshop Proceedings

1. **Mounika Marreddy**, Lakshmi Vakada Sireesha, Venkata Charan Chinni, Subba Reddy Oota, Radhika Mamidi "Multi-Task Text Classification using Graph Convolutional Neural Networks for Resource-Poor Language," in (WiML-NIPS)-2020. (Extended Abstract)

2. **Mounika Marreddy**, Subba Reddy Oota, Radha Agarwal, Radhika Mamidi, "Evaluating the Combination of Word Embeddings with Mixture of Experts and Cascading gcForest in Identifying Sentiment Polarity," in (KDD-WISDOM)-2019.

Work in progress

1. Lakshmi Vakada Sireesha, Anudeep Ch, Subba Reddy Oota, **Mounika Marreddy**, Radhika Mamidi,"GAE-ISumm": Unsupervised Graph-Based Summarization of Indian Languages.

Other Accepted Works

1. Subba Reddy Oota, Jashn Arora, Veeral Agarwal, **Mounika Marreddy**, Raju Bapi, Manish Gupta, "Neural Language Taskonomy: Which NLP Task is most Predictive of fMRI Brain Activity?," in (NAACL)-2022.

2. Suma Reddy, Subba Reddy Oota, **Mounika Marreddy**, Radhika Mamidi,"TeluguNER: Leveraging Multi-Domain Named Entity Recognition with Deep Transformers," in Association for Computational Linguistics (ACL-SRW)-2022.

3. Lakshmi Vakada Sireesha,Venkata Charan Chinni,**Mounika Marreddy**,Subba Reddy Oota, Radhika Mamidi,"Unsupervised Graph based Telugu News Articles Text Summarization" in (WiML-NeurIPS-2020) - (Extended Abstract).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Text Classification

Text classification is a well-studied problem in the field of NLP, which has important applications like sentiment analysis (SA) [92, 21, 129, 113], emotion identification (EI) [1, 122, 73, 7], hate-speech detection (HS) [68, 145, 47, 106], and sarcasm detection (SAR) [55, 54, 76, 69]. Understanding text's contextual and semantic representation is one of the main challenges in text classification. Also, the traditional methods represent text with hand-crafted features, including structural and part-of-speech (POS) tagging features, and depend on data availability.

Efficient feature representations are the cornerstones of NLP text classification applications. Particularly, the English language has many word embedding representations at contextual [97, 35] and non-contextual levels [80, 95]. These representations help in solving NLP tasks like sentiment analysis [92, 102], question answering [14, 132], and semantic role labeling [118, 27]. However, the quality of the embeddings is impacted by the size of the mono-lingual corpora [80, 13]. Also, the input representation for Indian languages is very sparse and unsatisfactory due to the absence of a large corpus and annotated datasets.

Most of the resources created for Indian languages are from a machine translation perspective. However, for other NLP tasks, the available resources are significantly fewer. Hence, creating abundant data, tools, and machine learning or deep learning models for Indian languages is the need of today as it will help increase communication and understanding of contextual difficulties. Further, these resources are helpful for NLP applications like text classification, text summarization, question answering, and automated document evaluation, which might motivate future endeavors in Indian languages like Telugu, the language selected for the current research work.

Recently, most social media, e-commerce, and news platforms have started allowing users to express their opinions and thoughts in their native language, as shown in Figure 1.1. Specifically, many applications and tools, including Google, Instagram, Facebook, and WhatsApp, provide native language selection and translation options. From Figure 1.1, we observe that people post

different types of responses online. For example, hateful comments on Facebook posts, sarcastic tweets on Twitter posts, sentimental and emotional comments on YouTube movie videos, and Telugu news articles. Moreover, on platforms like Facebook and Instagram, posts using Indian native languages get 50% more responses compared to English posts[1]. Hence, motivated by the immense use of such online platforms and apps, we investigate and use the available data for creating a large corpus, better-annotated datasets, models, and tools in Telugu.

The problem is challenging because creating such large corpora and annotated data requires standard annotation guidelines, manual power, cost, and sentence pre-processing, including ambiguous ones (lexical, semantic, and structural levels). In addition, the pre-processing of Indian languages is arduous from English because of the morphology and syntax. The second challenge is to verify the quality of the annotated data, as the data we created is multi-domain and taken from different sources, as mentioned above. Since anyone can create articles on Wikipedia, which sometimes may not be authentic, there is a need to verify the facts. We extracted the relevant Telugu data from other websites to make a better corpus to overcome this challenge. In addition to handling imbalanced datasets, verifying contexts for ambiguous sentences is challenging. Overall, we attempt to create a web-based annotation tool to simplify the annotation tasks for an end-user.

Some concerted and noteworthy efforts have been made to build the pre-trained word-embedding models for Telugu, including Word2Vec in [87], an n-gram-based skip-gram model to train the FastText embeddings in [13]. Similar to the n-gram approach, sub-word pre-trained word embeddings based on Byte-Pair Encoding (BPE) are trained on Telugu Wikipedia in [48]. Also, the recent successful multi-lingual pre-trained transformer models such as mBERT [34] and Cross-Lingual Language Model (XLM-R) [29] use Telugu corpus as one of the multi-lingual corpora for training. However, these models do not have high-quality representations for all languages. For example, mBERT performs worse than non-BERT models on several downstream tasks for the bottom 30 percent of languages in terms of dataset size upon which mBERT is trained [130]. The main drawback of these methods is that they are never tested on large Telugu annotated corpora, and no benchmark results are available for different NLP tasks in Telugu. Recently, [57] introduced a multi-lingual language model (pre-trained in many Indian languages), IndicBERT, to evaluate the downstream tasks of eleven major Indian languages. However, these models do not focus on a specific language and are particularly unwieldy due to their large number of parameters. Therefore, creating resources and building language models in the mono-lingual setting of our selected low-resource language, Telugu, is essential.

From this work, we want to bridge the gap by creating resources for Telugu. Examples of Telugu language sentences from our corpus, their WX notation (a standard notation used for Indian languages)[2], and English translation are reported in the Figure 1.2. From Figure 1.2, we observe that it is possible to create sentences having all kinds of conversations, opinions,

[1] `http://tiny.cc/1yuzpz`

[2] `https://en.wikipedia.org/wiki/WX_notation`

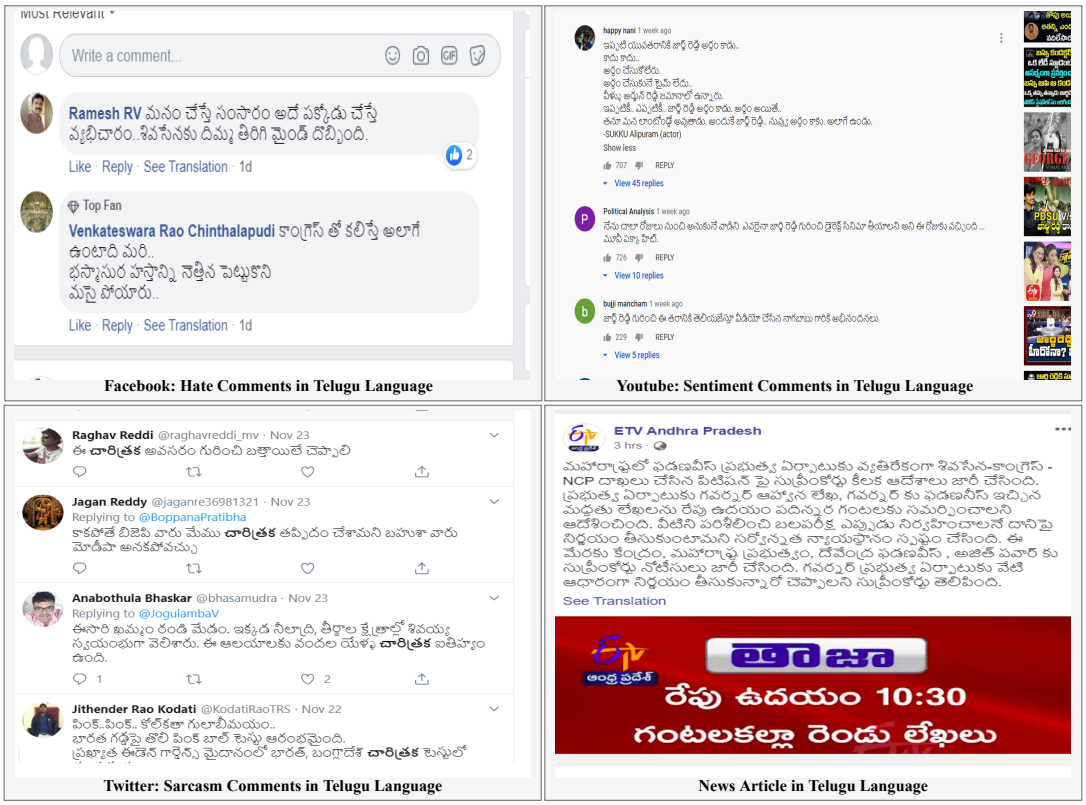


Figure 1.1: Sample of Telugu language comments on different social and news media platforms

and expressions and obtain valuable insights. Hence, creating copious data for multiple NLP tasks can overcome the limitations and help the researchers make the best models for Telugu.

1.2 Observations and Research Issues

We list some observations that overcome the fundamental limitations of Telugu NLP models and resources and present the research issues.

1.2.1 Observations

1. **Corpus Creation:** For the Telugu language, we do not have an available corpus to generate pre-trained embeddings. We are the first to create a large Telugu corpus of 80,15,588 sentences crawled from websites publishing Telugu data. We make the corpus publicly available.

2. **Lack of Annotated Datasets:** We create different large annotated datasets for the five NLP tasks in the Telugu language, in contrast to earlier works that experimented with tiny data for a single task [87]. We publicly released the five annotated datasets.

Sentence	Label
రేణిగుంట విమానాశ్రయంలో గవర్నర్ నరసింహన్, ఏపీ సీఎం జగన్ ప్రధానికి ఘనస్వాగతం పలికారు. reNiguMta vimAnASrayaMlo gavarnar narasiMhan, epI sIeVM jagan praXAniki GanasvAgawaM palikAru. Prime Minister was grandly welcomed by Governor Narasimhan and AP CM Jagan at Renigunta airport.	Sentiment: Positive Emotion: Happy
పశ్చిమబెంగాల్ లో అధికార తృణమూల్ హయాంలో గూండా రాజ్యం రాజ్యమేలుతోందని ప్రధాని మోడీ దుయ్యబట్టారు. paScimabeVMgAl lo aXikAra wqNamUl hayAMlo gUMdA rAjyaM rAjyameluwoMxani praXAni modI xuyyabattAru. Prime Minister Modi has lashed out that the apache government is in power in West Bengal.	Emotion: Angry Hate-speech: Yes
జగన్, విజయసాయిరెడ్డిలు అవినీతి బురదలో కూరుకుపోయిన మురికి వ్యక్తులని విమర్శించారో. jagan, vijayasAyireVddilu avinIwi buraxalo kUrukupoyina muriki vyakwulani vimarSiMcAru. Jagan and Vijayasai Reddy have been criticized for being bogged in the dirt of corruption.	Hate-speech: Yes Sarcasm: Yes

Figure 1.2: Example sentences with labels from different tasks in Telugu, WX, and English translation

3. **Absence of Efficient Feature Representations:** Creating copious data for multiple NLP tasks can overcome the limitations and help researchers make the best models for Telugu. We create fifteen different feature representations from scratch on 80,15,588 sentences. In particular, we create recent pre-trained language models for Telugu such as *ELMo-Te*, *BERT-Te*, *RoBERTa-Te*, *ALBERT-Te*, and *DistilBERT-Te* on our large Telugu corpora. These embeddings are made publicly available.

4. **Non-availability of Telugu Machine Learning Models:** Unlike English, very few preliminary works have covered text classification in Telugu. Unfortunately, all these works were limited to existing baseline models that use either small annotated datasets [86] or restricted to single task text classification [87]. Hence, we propose a multi-task graph convolutional network (MT-Text GCN) to jointly (i) perform graph reconstruction and (ii) do multi-task text classification on four significant NLP tasks - SA, EI, HS, and SAR. We also introduce linear probing and fine-tuning of Telugu pre-trained transformers and compare them with multi-lingual pre-trained transformers. We perform linear probing and fine-tuning on NLP tasks (SA, EI, HS, and SAR) and the available state-of-the-art Telugu datasets. We also did a complete end-to-end automated Telugu clickbait detection model.

1.2.2 Research Issues

Based on the preceding observations, we notice that no significant effort has been made to create a raw corpus, guidelines for annotation, annotated datasets, and models for the Telugu language. The main research issues identified are: to create a large corpus, annotated datasets for text classification, develop feature representations from scratch, and build machine or deep learning models for Telugu.

1.3 Research Methodology

A research method is a set of instructions for solving a problem. The common components of a research method are stages, tasks, methods, techniques, and tools. Research methods are broadly classified into the scientific method and the mathematical method. Scientifically, the first step is to observe the world, propose a theoretical model of behavior, measure and analyze it, confirm the model's hypothesis or theory, and repeat the process if possible. The mathematical method proposes a formal theory or a set of axioms, develops the idea, derives the results, and compares it with empirical observations. The scientific method can be divided into two models: Engineering and empirical.

Engineering method: Here, the existing solutions were observed, and better solutions were proposed, designed, and developed. The evaluation metrics were defined to measure and analyze the solutions, and the process is repeated until no more improvements are possible.

Scientific method: Here, models are proposed, and efforts were made to develop statistical and quantitative methods. These models are applied to the case studies to measure and analyze the model validation. The process is repeated for a few iterations to develop a standard model.

The proposed approach in this thesis comes under the category of the engineering method. We first observed existing text classification methods for Telugu and identified the drawbacks. We created the resources, datasets, and tools for the text classification tasks in Telugu. However, these help to build feature representations and models in Telugu. We made an effort in the Telugu language and developed the corpus, annotated data, machine learning, and deep learning resources.

1.4 Overview of the Thesis

In this thesis, as the initial works, we experimented with available feature representation methods in the English language. Also, we proposed a mixture of the expert model to evaluate the quality of word embeddings. By analyzing the results of the feature representations in English, we feel that for the success of NLP in any language, one needs to develop the feature representations from scratch. We created a corpus and made 18 feature representations from scratch in the Telugu language. We have created various Telugu annotated datasets for multiple NLP text classification tasks in Telugu. We proposed graph-based architectures, machine learning, and deep learning-based models. We compared our results with existing multi-lingual representations and got the best results with the created representations. The overview of the problem framework and the proposed architectures are as follows:

1.4.1 About the Problem Statement

More than 6000 languages are spoken worldwide, yet the available Speech and NLP datasets only cover a limited number of languages, usually a few hundred. Consequently, the performance of the best Speech and NLP models is primarily optimized for a small subset of languages. Researchers have been focusing on a multi-lingual approach in recent years, although it has been challenging to encompass even a hundred languages within this framework.

Relying on a paradigm that builds model architectures primarily based on English, French, Spanish, and a few other European languages, with the expectation that these models will generalize effectively to all 6000 languages, has proven to be ineffective. Our qualitative and quantitative experiments in Telugu (a mono-lingual context) have shown that creating pre-trained models from scratch yields superior results compared to existing multi-lingual models.

Moreover, there is a pressing need to develop language-specific models for various languages and language families worldwide. By combining the insights gained from these individual models, we can strive towards creating comprehensive multi-lingual models. This thesis addresses the challenge of the low-resource problem of Telugu language NLP processing by building a large corpus, multiple annotated datasets, and pre-trained architectures from scratch. Machine learning and deep learning models specifically designed for Telugu were also developed in this research endeavor.

1.4.2 Understanding the Intensity of Tweets using Feature Representations

The main goal of our first work is to understand the importance of feature representations in a given NLP task (i.e., Estimating the intensity of emotion) using English data. Estimating the intensity of emotion has gained significance as modern textual inputs in potential applications like social media, e-retail markets, psychology, advertisements, etc., carry a lot of emotions, feelings, and expressions along with their meaning. The intensity of emotion from text has emerged as an increasingly important and standardizing modern textual input in potential applications like social media, e-retail markets, psychology, advertisements, etc. However, traditional sentiment analysis approaches primarily focus on classifying the sentiment in general (positive or negative) or at an aspect level (very positive, low negative, etc.) and cannot exploit the intensity information. Moreover, automatically identifying emotions like anger, fear, joy, sadness, disgust, etc., from text introduces challenging scenarios where a single tweet may contain multiple emotions with different intensities and some emotions may co-occur in some tweets. This work proposes an architecture, the Experts Model, inspired by the standard Mixture of Experts (MoE) model. The key idea here is each expert learns different sets of features from the feature vector which helps in better emotion detection from the tweet. We compared our Experts Model's results with baseline results and the top five performers of SemEval-2018 Task-1, Affect in Tweets (AIT). The experimental results show that our proposed approach

deals with the emotion detection problem and stands at the top-5 results. The detailed work is presented in Chapter 3.

1.4.3 Evaluating the Feature Representations

The following work with English data evaluates the existing pre-trained embeddings using a mixture of expert model. The main aim of this work is to see which feature representation is important for a particular NLP task (in this case, I experimented with sentiment analysis). Neural word feature embeddings have delivered impressive results in many Natural Language Processing tasks. The quality of the word embedding determines the performance of a supervised model. However, choosing the right word embeddings for a given dataset is a primary challenge for enhancing the results. In this paper, we have evaluated neural word embeddings on sentiment analysis task in two steps: (i) propose a mixture of classification experts (MoCE) model for the sentiment classification task, (ii) compare and improve the classification accuracy by different combination of word embedding as the first level of features and pass it to cascade model inspired by gcForest for extracting various features. We argue that each expert learns certain positive or negative examples corresponding to its category in the first step. In the second step, resulting features on a given task (polarity identification) can achieve competitive performance with state-of-the-art accuracy methods, precision, and recall using gcForest. The detailed work is presented in Chapter 4.

1.4.4 General Annotation Schema for NLP Tasks in Telugu

From here, we focus on the primary motivation of the thesis, i.e., to mitigate the low resource problem of the mono-lingual language (Telugu). In this chapter, we explained the process of developing a large annotated work, the annotation process, and the NLP tasks we considered in Telugu. A detailed explanation is presented in Chapter 5.

1.4.5 TEL-NLP Dataset and Multi-task GCN for Text Classification

In this work, we study the use of GCN for the Telugu language in single and multi-task settings for four natural language processing (NLP) tasks, viz. sentiment analysis (SA), emotion identification (EI), hate-speech (HS), and sarcasm detection (SAR). To evaluate the performance of GCN with one of the Indian languages, Telugu, we analyze the GCN-based models with extensive experiments on four downstream tasks. In addition, we created an annotated Telugu dataset, TEL-NLP, for the four NLP tasks. Further, we propose a supervised graph reconstruction method, Multi-Task Text GCN (MT-Text GCN) on the Telugu that leverages to simultaneously (i) learn the low-dimensional word and sentence graph embeddings from word-sentence graph reconstruction using graph autoencoder (GAE) and (ii) perform multi-task text

classification using these latent sentence graph embeddings. We argue that our proposed MT-Text GCN achieves significant improvements on TEL-NLP over existing Telugu pre-trained word embeddings [78], multi-lingual pre-trained Transformer models. The detailed work is presented in Chapter 6.

1.4.6 Datasets, Embeddings, and Models for Text Classification Tasks

In this work, we explore the traditional to recent efficient representations to overcome the challenges of low resource language, Telugu. In particular, our main objective is to mitigate the low-resource problem for Telugu. Overall, we present several contributions to a resource-poor language, viz. Telugu. (i) a large annotated data (35,142 sentences in each task) for multiple NLP tasks such as sentiment analysis, emotion identification, hate-speech detection, and sarcasm detection, (ii) we create different lexicons for sentiment, emotion, and hate-speech for improving the efficiency of the models, (iii) pre-trained word and sentence embeddings, and (iv) different pre-trained language models for Telugu such as *ELMo-Te*, *BERT-Te*, *RoBERTa-Te*, *ALBERT-Te*, and *DistilBERT-Te* on a large Telugu corpus consisting of 80,15,588 sentences (16,37,408 sentences from Telugu Wikipedia and 63,78,180 sentences crawled from different Telugu websites). Further, we show that these representations significantly improve the performance of four NLP tasks and present the benchmark results for Telugu. We argue that our pre-trained embeddings are competitive or better than the existing multi-lingual pre-trained models. Lastly, the fine-tuning of pre-trained models shows higher performance than linear probing results on four NLP tasks. We also experiment on publicly available Telugu datasets (Named Entity Recognition, Article Genre Classification, and Sentiment Analysis), find that our Telugu pre-trained language models (*BERT-Te* and *RoBERTa-Te*) outperform the state-of-the-art system except for the sentiment task. The detailed work is presented in Chapter 7.

1.4.7 Dataset, Embeddings, and Models for Clickbait Detection

In this work, we present an annotated clickbait dataset of 112,657 headlines that can be used for building an automated clickbait detection system for Telugu, a resource-poor language. We show that the pre-trained language models trained on Telugu outperform the existing pre-trained models on clickbait task. On a large Telugu clickbait dataset of 112,657 samples, the Light Gradient Boosted Machines (LGBM) model achieves an F1-score of 0.94 for clickbait headlines. For Non-Clickbait headlines, an F1-score of 0.93 is obtained, similar to that of the Clickbait class. The detailed work is presented in Chapter 8.

1.5 Contributions of the Thesis

As mentioned earlier, there is a need to create a corpus, annotate datasets and embeddings, and develop machine learning and deep learning models to understand contextual predictions and decisions better. To our knowledge, no such resources are available for Telugu, and we are the pioneers in creating a large corpus of copious labeled Telugu data for five NLP tasks sentiment, emotion, hate-speech, sarcasm, clickbait, and pre-trained language models. The main contributions of this thesis are as follows:

- We created a large Telugu corpus of 80,15,588 sentences (16,37,408 sentences from Telugu Wikipedia and 63,78,180 sentences crawled from different Telugu websites).
- We created annotated datasets for sentiment analysis, emotion identification, hate speech detection, sarcasm identification, and clickbait detection.
- For the Telugu language, we are the first to create lexicons, Word2Vec, GloVe, FastText, Skip-Thought vectors, pre-trained language models for Telugu such as *ELMo-Te*, *BERT-Te*, *RoBERTa-Te*, *ALBERT-Te*, and *DistilBERT-Te*, word embeddings using random walk-based models: DeepWalk, Node2Vec, and Graph AutoEncoders (GAE) from scratch on the created Telugu corpus.
- We propose the multi-task learning model (MT-Text GCN) to reconstruct word-sentence graphs while achieving multi-task text classification with learned graph embeddings.
- We outperformed the multilingual pre-trained language models such as *mBERT* [34], *XLM-R* [70], *Sub-word Byte Pair* [48], *Facebook FastText* [13], and *IndicBERT* [57].

The sentence-level corpora, annotated datasets, lexicons, pre-trained words, sentence embeddings, and pre-trained transformer models are publicly available under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. We open-source our corpus, four different datasets, lexicons, embeddings, and code[3]. The pre-trained transformer models for Telugu are available here[4].

1.6 Organization of the Thesis

The rest of the thesis is organized as follows:

- **Chapter 2. Related work:** In this chapter, we discuss the literature survey of existing datasets in Telugu and the different feature representations available for English and about the NLP tasks.

[3]https://github.com/mounikamarreddy/NLP-for-Telugu-Language.git
[4]https://huggingface.co/ltrctelugu

- **Chapter 3. Evaluating Word representations using Sentiment Analysis:** This is the first work to understand how feature representations are essential for sentiment analysis. We have evaluated neural word embeddings on sentiment analysis task in two steps: (i) proposed a mixture of classification experts (MoCE) model for the sentiment classification task, (ii) to compare and improve the classification accuracy by different combinations of word embedding as the first level of features and pass it to cascade model inspired by gcForest for extracting various features.

- **Chapter 4. Affects in Tweets using Experts Model:** In this work, we propose an architecture, the Experts Model, inspired by the standard Mixture of Experts (MoE) model. The key idea here is each expert learns different sets of features from the feature vector, which helps in better emotion detection from the tweet. We compared our Experts Model's results with baseline results and the top five performers of SemEval-2018 Task-1, Affect in Tweets (AIT).

- **Chapter 5. General Annotation Schema for Five NLP Tasks in Telugu:** In this chapter, we present the corpus creation, annotation process, annotation guidelines with Telugu examples, the tool we created to ease the annotation process, and fair compensation to the annotators.

- **Chapter 6. TEL-NLP Dataset and Multi-task GCN for Text Classification:** In this chapter, we present an annotated TEL-NLP dataset in Telugu covering four NLP tasks (16,234 samples for SA, HS, SAR, and 9,675 samples for EI). We propose a supervised graph reconstruction method, Multi-Task Text GCN (MT-Text GCN), for the Telugu language in single and multi-task settings for four natural language processing (NLP) tasks.

- **Chapter 7. Datasets, Embeddings, and Models for Text Classification Tasks:** In this chapter, our main objective is to mitigate the low-resource problem for Telugu. We present an annotated dataset of 35,142 sentences for four NLP tasks. We developed machine learning models that automatically detect the task outputs using traditional to recent successful feature representations built from scratch.

- **Chapter 8. Dataset, Embeddings and Models for Clickbait Detection:** In this chapter, we present an annotated clickbait dataset of 112,657 headlines that can be used for building an automated clickbait detection system for the Telugu language.

- **Chapter 9. Summary, Conclusions, and Future Work:** In this chapter, we present the thesis's summary, conclusions, limitations, and future scope.

Chapter 2

Related Work

In this chapter, we present the related work for feature representation methods and the literature survey on the existing NLP tasks for English language.

2.1 Feature Representation Methods

Just as in other types of machine learning tasks, we must find a way to represent our data (a series of texts) to our systems (e.g., a text classifier) in NLP. This representation should be an efficient representation of the text, capture the context-related information, and validate it to do math operations, which, in turn, captures a semantic representation of the text. In this section, we will be looking at feature engineering strategies that often leverage tradition and advanced deep learning models. Specifically, we will be covering the lexicons, BoW, TF-IDF, Word2Vec, GloVe, FastText, Skip-Thought, ELMo, and BERT models as described below.

2.1.1 Lexicon Based Features

The lexicon-based approach involves calculating orientation for a document from the semantic orientation of words, phrases, or word senses in the document. Generally, in automatic sentiment analysis, dictionaries annotated with words sentiment orientation are used. These dictionaries can be created manually using the main word in the sentence, which exhibits the sentiment or semi-automatically [50] using existing resources like WordNet or automatically [124] by using seed words and then expanding the list. Lexicons [61, 128] have leveraged as useful and successful feature for sentiment analysis. These results provide a word-level foundation for identifying sentiment or emotion in sentences and documents.

2.1.2 Traditional Feature Representation Methods

The two traditional feature representation methods Bag-Of-Words (BoW) [126], and TF-IDF [105] are successful in many of the NLP tasks. Although these methods are simple to

implement, yet produce better results. In the next subsections, we discuss these two traditional feature extraction methods.

2.1.2.1 Bag-of-Words (BoW)

A bag-of-words (BoW) is a representation of text that describes the occurrence of words within a document. In this approach, we use the tokenized words for each observation and build the vocabulary by identifying the unique words in the corpus. Once the vocabulary file is built, we need to vectorize our documents by counting how many times each word appears. The intuition is that documents are similar if they have similar content. Further, from the content alone, we can learn something about the meaning of the document. The complexity of the BoW method comes both in deciding how to design the vocabulary of known words (or tokens) and how to score the presence of known words.

2.1.2.2 TF-IDF

To overcome the limitation of highly frequent words with no informational content, an approach named Term frequency-Inverse document frequency (TF-IDF) [105] is one such model used to rescale the frequency of words by how often they appear in all documents so that the scores for frequent words across all documents are penalized. TF-IDF weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

- Term Frequency: is a scoring of the frequency of the word in the current document.
- Inverse Document Frequency: is a scoring of how rare the word is across documents.

$$W_{i,j} = tf_{i,j} * log(\frac{N}{df_i}) \tag{2.1}$$

Here $tf_{i,j}$ is the number of occurrences of i in j, df_i is the number of documents containing i and 'N' is the total number of documents.

2.1.2.3 Words Co-Occurrence Features

Words co-occurrence statistics describe how words occur together that, in turn, captures the relationships between words. Words co-occurrence statistics are computed simply by counting how two or more words occur together in a given corpus. Co-occurrence matrices capture a lot of the information you need for natural language processing (NLP). They contain both semantic and syntactic information. Also, terms with related meanings (a semantic relationship) often have high co-occurrence with each other (for example, sporting terms tend to cluster together).

Words with a similar syntactic role often have similar co-occurrence patterns (for example, in English, verbs often co-occur with the word "to").

2.1.2.3.1 Co-Occurrence Matrix Construction Given a corpus of N words, we need a table of size NxN to represent bigram/trigram/n-gram (depends on window size) frequencies of all possible word-pairs. Such a table is highly sparse as most frequencies are equal to zero. However, this co-occurrence matrix is not the word vector representation that is generally used. Instead, this Co-occurrence matrix is decomposed using techniques like PCA (Principle Component Analysis), SVD (Singular Value Decomposition), etc. into factors, and a combination of these factors forms the word vector representation.

2.1.3 Word Embedding Representations

Word embedding methods allow the words with similar meaning understood by machine learning algorithms. Moreover, these distributed word representations capture a large number of precise syntactic and semantic word relationships. In this section, we describe different word embeddings, such as Word2Vec, GloVe, and FastText.

2.1.3.1 Word2Vec

2.1.3.1.1 Shortcomings of Earlier Approaches Traditional feature representation methods such as BoW, TF-IDF, ignores the order of the word; for example, this is bad = bad is this. Another feature method LSA (Latent Semantic Analysis), used the concept of Bag-of-words, where words are represented in the form of encoded vectors. It is a sparse vector representation where the dimension is equal to the size of the vocabulary. If the word occurs in the dictionary, it is counted, else not. Moreover, all the above-mentioned methods ignore the context and semantics of words.

2.1.3.1.2 Contributions of Word2Vec

- Word2Vec [80] introduce techniques that can be used for learning high-quality word vectors from huge data sets with billions of words, and with millions of words in the vocabulary.
- In Word2Vec, the word representations are dense and having reduced dimension (depends on the number of hidden neurons).
- Try to maximize the accuracy of word vector operations by developing new model architectures that preserve the linear regularities among words.
- Design a new comprehensive test set for measuring both syntactic and semantic regularities, and show that many such regularities can be learned with high accuracy.

- Two new model architectures proposed for learning distributed representations of words that try to minimize computational complexity.

2.1.3.1.3 Model Architectures Word2Vec is not a single algorithm but a combination of two techniques – CBOW(Continuous Bag Of Words) and SG (Skip-Gram) model. Both of these are shallow neural networks that map word(s) to the target variable, which is also a word(s). Both of these techniques learn weights that act as word vector representations.

- The objective of CBOW is to predict the current word based on neighboring words.
- The objective of SG is to predict the neighboring words based on window size given the current word.

The training objective of the Skip-Gram model is to find word representations that are useful for predicting neighboring words in a sentence or document.

Let w_1, w_2, w_3,.....,w_T represent the sequence of training words. The objective of SG is to maximize the log probability and is given by the below equation:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(\frac{w_{t+j}}{w_j}) \tag{2.2}$$

where c is the training context size. The SG formulation defines $p(w_{t+j}/w_j)$ using the below softmax function:

$$p(w_O/w_I) = \frac{\exp(v'^T_{w_O} v_{w_I})}{\sum_{w=1}^{W} \exp(v'^T_w v_{w_I})} \tag{2.3}$$

where v_w and v'_w are the "input" and "output" representations of w, and W is the number of words in the vocabulary.

2.1.3.2 GloVe

Another well-known model that learns vectors or words from their co-occurrence information, i.e., how frequently they appear together in large text corpora, is Global Vectors (GloVe) [95]. While word2Vec is a predictive model — a feed-forward neural network that learns vectors to improve the predictive ability, GloVe is a count-based model.

2.1.3.2.1 Contributions

- The main idea of GloVe [95] is to utilize the statistics of the whole corpus.
- GloVe takes global information into account while learning the dimensions of meaning.
- GloVe model efficiently uses statistical information by training on a word-word co-occurrence matrix rather than on the entire sparse matrix or individual context windows in a large corpus.

2.1.3.2.2 Model Architecture The way GloVe predicts surrounding words is by maximizing the probability of a context word occurring given a center word by performing a dynamic logistic regression. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning.

Given that i and j are two words that co-occur, we can optimize a word vector by minimizing the difference between the dot product of the word vectors for i and j, and the log of the number of times i and j co-occur, squared, and the objective function of the glove is given by the below equation:

$$\hat{J} = \sum_{i,j}^{W} f(X_{i,j})(w_i^T \tilde{w}_j - logX_{ij})^2 \tag{2.4}$$

where $\hat{J}$ is the objective function, W is the size of vocabulary, $w_i^T w_j$ is the dot product of vectors w_i and w_j, X_{ij} is the count of the number of times i and j co-occur. Here $f(X_{ij})$ is the weighing function to avoid undefined value of objective function when i and j never co-occur (X_{ij}=0 and log 0 is undefined).$f(X_{ij})$ has a few properties. First, $f(X_{ij}) = 0$ when $X_{ij} = 0$. This means that when i and j don't co-occur, you don't need to calculate $(uiTvj - logPij)^2$, we can just stop at $f(X_{ij})$. Secondly, $f(X_{ij})$ helps counteract the problem of balancing the weight of very common or very uncommon words.

2.1.3.3 FastText

The word embeddings trained using Word2Vec, and GloVe models, ignore the internal structure of the words, represent a distinct word vector. This is the main drawback of morphologically abundant languages. Moreover, these models do not provide vectors for rare or out-of-vocabulary words. To overcome the above limitations, the authors introduced a FastText model in [56], an extension of the Word2Vec model. Since FastText model considers the bag of character n-grams to represent each word [56], it allows us to compute rare word representations as well. The objective function of FastText is to minimize the below equation:

$$\frac{-1}{N} \sum_{n=1}^{N} y_n log(f(BAx_n)) \tag{2.5}$$

where x_n is the normalized bag of features of the n-th document, y_n is the label, A and B the weight matrices.

FastText supports training continuous bag of words (CBOW) or Skip-gram models using negative sampling, softmax or hierarchical softmax loss functions.

2.1.4 Sentence Embedding Representations

In literature, several researchers use deep learning approaches for learning sentence vectors that mapped from word vectors such as recurrent networks, recursive networks, convolutional networks, and recurrent convolutional networks. All these methods learn the representation of a sentence in a supervised fashion, i.e., depends on the class label. As a result, the representations of a sentence are of high quality, but specific to the respective tuned task.

Motivated by the recent successful language models which try to learn the sentence vectors in an unsupervised fashion, we discuss the Skip-Thought vectors.

2.1.4.1 Skip-Thought Vectors

Skip-Thought model is a sentence encoding-decoding model that produces a fixed-length vector for every given sentence [67]. Inspired by the Word2Vec approach, the skip-thought model extends the skip-gram model to generate sentence embeddings from word embeddings. Rather than predicting the context given the input word, the skip-thought model predicts both the next and previous sentences given the target sentence.

The skip-thought model have three parts:

- **Encoder:**Takes the sentence x(i) at index i and generates a fixed-length representation z(i). This is a recurrent network with GRU activation that takes the words in a sentence sequentially.

- **Previous Decoder Network:** Takes the embedding z(i) and "tries" to generate the sentence x(i-1). This also is a recurrent network with conditioned GRU that generates the sentence sequentially.

- **Next Decoder Network:** Takes the embedding z(i) and "tries" to generate the sentence x(i+1). Again a recurrent network similar to the Previous Decoder Network.

2.1.4.1.1 Skip-Thought Model Training Details

- Skip-thought assumes that when a sentence leads to a better reconstruction of neighboring sentences, then that particular sentence is the main essence.

- The decoders are trained to minimize the reconstruction error of the previous and the next sentences given the embedding z(i).

- This reconstruction error is back-propagated to the Encoder, which now has a "motivation" to pack as much information about sentence x(i) that will help the decoders minimize the error in generating the previous and next sentences.

- The end product of Skip-Thoughts is the encoder, which is used to generate fixed-length representations of sentences that can be used for several downstream tasks such as sentiment classification, semantic similarity, etc.

- In skip-thought model given a tuple (s_{i-1}, s_i, s_{i+1}) of contiguous sentences, with s_i the i-th sentence of a book, the sentence s_i is encoded and tries to reconstruct the previous sentence s_{i-1} and next sentence s_{i+1}.

Given a tuple (s_{i-1}, s_i, s_{i+1}), the objective optimized is the sum of the log-probabilities for the forward and backward sentences conditioned on the encoder representation and is given by the below equation:

$$\sum_t logP(w_{i+1}^t | w_{i+1}^{<t}, h_i) + \sum_t logP(w_{i-1}^t | w_{i-1}^{<t}, h_i) \tag{2.6}$$

2.1.5 Contextual Word Representations

2.1.5.1 ELMo (Deep contextualized word representations)

ELMo (Embeddings from Language Model) [97] overcomes the limitations of traditional word embedding methods by learning contextualized word embeddings from the language model.

2.1.5.1.1 Limitations with Previous Methods Traditional NLP techniques and frameworks were great when asked to perform basic tasks. Traditional word embeddings such as Word2Vec, GloVe, and FastText come up with the same vector for the word in both the sentences. Hence, the system would fail to distinguish between the polysemous words. These word embeddings just cannot grasp the context in which the word was used.

2.1.5.1.2 Contributions of ELMo

- Embeddings from Language Models (ELMos) use language models to obtain embeddings for individual words while taking the entire sentence or paragraph into account.

- Concretely, ELMos use a pre-trained, multi-layer, bi-directional, LSTM-based language model and extract the hidden state of each layer for the input sequence of words. Then, they compute a weighted sum of those hidden states to obtain an embedding for each word.

- ELMo improves the performance of models across a wide range of tasks, spanning from question answering and sentiment analysis to named entity recognition.

With out labelling data we can train vast amounts of running text. Given a sequence of N tokens, (t_1, t_1, ..., t_N), the forward language model computes the probability of sequence by modelling the probability if token t_k given the history (t_1, t_1, .., t_{k-1}):

$$p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, .., t_{k-1}) \tag{2.7}$$

The backward language model predicts the previous token given the future context and is given by the below equation:

$$p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{K+2}, .., t_N) \tag{2.8}$$

A biLM combines both forward and backward language models. The objective function is to maximize the log likelihood of the forward and backward language models and is given by the below equation:

$$\sum_{k=1}^{N} (logp(tk | t_1, .., t_{k-1}; \Theta_x, \Theta_{\overrightarrow{LSTM}}, \Theta_s) + logp(tk | t_{k+1}, .., t_N; \Theta_x, \Theta_{\overleftarrow{LSTM}}, \Theta_s)) \tag{2.9}$$

where Θ_x, Θ_s are the token representation and softmax layer parameters. ELMo equation for task specific representation is given by the below equation:

$$ELMo_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}^{LM} \tag{2.10}$$

where γ is of practical importance to aid the optimization process and γ^{task} allows the task model to scale the entire ELMo vector, s_j^{task} are softmax normalized weights, $h_{k,j}^{LM}$ are top layer biLSTM outputs.

The training process of ELMo is as follows:

1. First, we built a vocabulary file that contains all the unique words along with three extra tokens <S>, </S>, </UNK>.
2. The vocabulary file is in the descending order where the most frequently occurring word in the top.
3. A data file that contains all the sentences in the corpus where one sentence per line.
4. We split the data into different files in which 70% of the data we considered for training, remaining data held-out for testing.
5. A configuration file consists of the vocabulary size, number of tokens to train the ELMo model, output dimension of each token.
6. Each embedding is of dimension 1024.

2.1.5.2 BERT (Pre-training of Deep Bidirectional Transformers for Language Understanding)

BERT [33] is a deep learning model that has given state-of-the-art results on a wide variety of natural language processing tasks. It stands for Bidirectional Encoder Representations for Transformers.

2.1.5.2.1 Problem with Previous Methods Language models only use left context or right context, but language understanding is bidirectional. The feature-based approach, such as ELMo [97], uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) [104], introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning all pre-trained parameters. The major limitation of these techniques is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training.

2.1.5.2.2 Contributions of BERT

- BERT uses masked language models to enable pre-trained deep bidirectional representations. In masked language modeling instead of predicting every next token, a percentage of input tokens is masked at random, and only those masked tokens are predicted.
- BERT is also trained on a next sentence prediction task to better handle tasks that require reasoning about the relationship between two sentences (e.g., question answering).
- BERT is the first fine-tuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many task-specific architectures.

2.1.5.2.3 BERT Model Framework There are two steps in the BERT framework: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For fine-tuning, the BERT model is first initialized with the pre-trained parameters and all of the parameters are fine-tuned using labeled data from the downstream tasks.

BERT uses the transformer architecture (the attention mechanism that learns contextual relationships between words in a text) for its underlying model. A basic transformer consists of an encoder to read the text input and a decoder to produce a prediction for the task. Since the BERT goal is to generate a language representation model, it only needs the encoder part. The input to the encoder for BERT is a sequence of tokens, which are first converted into vectors and then processed in the neural network.

There are four types of pre-trained versions of BERT depending on the scale of the model architecture: BERT-Base: 12-layer, 768-hidden-nodes, 12-attention-heads, 110M parameters BERT-Large: 24-layer, 1024-hidden-nodes, 16-attention-heads, 340M parameters

2.1.5.2.4 Input/Output Representations

- Token embeddings: A [CLS] token is added to the input word tokens at the beginning of the first sentence, and a [SEP] token is inserted at the end of each sentence.
- Segment embeddings: A marker indicating Sentence A or Sentence B is added to each token. This allows the encoder to distinguish between sentences.
- Positional embeddings: A positional embedding is added to each token to indicate its position in the sentence.

2.2 Text Classification Tasks in English Language

In this section, we focus on five major popular NLP tasks, such as Sentiment Analysis, Emotion Identification, Hate-Speech Detection, Sarcasm Detection, and Clickbait Detection. We describe each one of the tasks below:

2.2.1 Sentiment Analysis

Sentiment analysis is one of the most popular tasks in the field of NLP. It helps in identifying and analyzing the subject information present in a user-written text [92, 102, 21, 139, 112]. Moreover, it is a well-established task in many business applications like movie reviews can cause or damage the revenue of the movie, and product reviews can encourage the quality of a product or damage the product sales. In literature, sentiment analysis performed on either sentence level or document level provides personal information to a user about his/her opinion [102, 123, 120, 136]. Further, with the recent developments of efficient deep feature representations yields better accuracies and tools for the English language. However, the amount of research work on sentiment analysis in Telugu is less compared to resource-rich language English due to the dearth of qualitative tools in the Telugu language. For sentiment analysis, the model should automatically predict the given text as “positive”, “negative”, or “neutral”.

2.2.2 Emotion Identification

People communicate their expressions, feelings, and thoughts on different topics in social media using emotional words. In the literature for the resource-rich language English, emotion identification has been applied to different domains including social media mining [116],

literature analysis [107], lyrics and music analysis [79], and the analysis of emoticons using deep moji [40]. Like Sentiment Analysis, Emotion Identification in textual conversations focuses on mapping words, sentences, or documents to emotion categories such as "joy", "sadness", "anger", "fear", "trust", "disgust", "anticipation", and "surprise", based on the models of [38, 99].

2.2.3 Hate-speech Detection

With the increase of social media platforms, the online content proliferates indefinitely, so does the rampant of hate-speech. Since users can express their freedom of speech at social media platforms causes the spreading of abusing information about other users [1]. Moreover, we can observe the hate-speech at different levels of users (i) local communities, (ii) members of different organizations (iii) users at the national level (terror attacks). As such, the content present in social media platforms like Twitter, YouTube, Facebook is harmful; the automatic removal of hate-speech content circumvent the spreading of hate-speech information. Many researchers are working from the past decade for automated hate-speech detection in the English language [68, 145, 127, 90]. For hate-speech, the model should automatically predict the given text as "yes" or "no".

2.2.4 Sarcasm Detection

With the enormous growth of the social media web, allow users to display their intent in the form of offensive or figurative language, including sarcasm, irony, humor, in public [8, 54]. Like sentiment and emotion text, sarcasm text should not be interpreted literally; it requires a more sophisticated understanding based on associations with the context or world knowledge. Moreover, the sarcasm task is often identified as a trope whose actual meaning differs from what is literally enunciated. Due to its nature, understanding sarcasm also emerges in the machine learning community. With the recent deep learning language models, the sarcasm task was successful in resource-rich English language [55, 142, 3]. For Sarcasm detection, the model should automatically predict the given text as "yes" or "no".

2.2.5 Clickbait Detection

With the impact of clickbait headlines in social media platforms, research on clickbait detection has addressed the issues of data creation, feature representation, and model building to automate the solution. The earlier works in the literature relied on creating annotated datasets and a rich set of hand-crafted features to categorize headlines as clickbait or non-clickbait [18]. Similar to the work mentioned above, there are a publicly available clickbait dataset developed

[1] `https://twitter.com/search?q=%23racism&src=typeahead_click`

on Twitter by considering the news genre, tweets from Twitter, and importance-based acquisition in [101], the headline from tweets, and gatekeeper-based acquisition in [2]. Here, [101] uses common words and extracted some other tweet specific features to classify tweets as clickbait or not. Furthermore, in [12], the authors described and constructed a new clickbait dataset by choosing the eight types of clickbait headlines. The main limitation of these works are that they focus on using hand-crafted features rather than automated ones to categorize the clickbait news headlines. For clickbait detection, the model should automatically predict the sentence as "yes" or "no".

2.3 Text Classification Tasks for Telugu Language

In the field of NLP, as discussed above, the well-established tasks in the English are sentiment analysis, emotion identification, hate-speech detection, sarcasm detection, and Clickbait. Unlike English, Indian languages are resource-poor due to the absence of corpus, resources, and tools. Also, most of the resources created for Indian languages are for machine translation. One of the solution is to translate existing English datasets for different NLP tasks in to Indian languages. However, by doing translation the, contextual meaning, syntactic and semantic structures will be affected and result in less accuracy [100]. Hence, creating abundant data, tools, and machine/deep learning models for Indian languages is the need of today as it will help increase communication and understand contextual difficulties. Further, these resources are helpful for NLP applications like sentiment analysis, text summarization, question answering, and automated document evaluation, which might motivate future endeavors in Telugu and other Indian languages.

The literature survey manifests that several authors worked on limited samples of Telugu data to solve the SA task. In particular, the earlier works in the literature focuses on creating sentiment-specific words and use a dictionary-based approach by considering the translated English sentiment words in SentiWordNet [31]. Similar to the work mentioned above, Telugu WordNet developed in [5, 88] using Hindi synsets. These works focus on translating from one language to another rather than directly creating words that are significant in understanding the context. Recently as an extension to the works mentioned above, (OntoSenseNet) [94], and BCSAT [93] created sentiment specific words for Telugu. However, the dataset is too small and limited to word-level meaning.

2.4 Deep Learning for Multi-task Text Classification

There is a vast literature tackling the single-task text (ST-Text) classification using deep learning models such as CNNs [62, 143] or RNNs [74]. However, these ST-Text classification approaches increase costs to build the resources and fail to combine multiple tasks. Multi-task

learning can improve the classification performance of related tasks by learning these tasks in parallel [17]. The literature survey shows that multi-task learning has been proven effective in solving similar NLP text classification problems [74, 131].

In the current research, graph neural networks has attracted wide attention [10], and applied on various NLP tasks such as text classification [121, 134], semantic role labeling [77], and machine translation [9]. The earlier works in text classification using GCN focused with nodes being either words or documents to construct a single large graph [32]. Some authors construct a single graph by jointly embedding the documents and words as nodes (Text GCN) [134] while considering the global relations between documents and words. The main drawback of the existing GCN works is that they cannot capture the contextual word relationships within the document. [144] developed a model, TextING, for inductive text classification to overcome the above limitations. [65] introduced GCN to embed the graph structure in a lower embedding space using neural networks. Motivated by [64], [66] propose a graph AutoEncoder framework in chapter-6 for Telugu that uses GCN as an encoder to get the latent representation of each node and simple inner product as a decoder to reconstruct the graph.

2.4.1 Sentiment Analysis in the Telugu Language

Despite enormous research that happened in multiple tasks of NLP for English, only a few authors worked on SA for Telugu [87, 94, 42]. In [87, 86, 42], authors provided Telugu SA data from different domains, categorized into three classes as "positive", "negative", and "neutral". However, the authors considered only 2X (2 persons) annotations (not sufficient for valid data), and these limited samples were used for model creation. In [42], authors did not report the features used for training the model, which does not make any valid comparison. Recently, [24] developed a deep learning-based model for Telugu SA to learn representations of resource-poor languages by jointly training with English corpora using a Siamese network. In these works, the model is trained on the English-Hindi corpus, and they apply transfer learning on the Telugu corpus, resulting in low accuracy. Several authors [125, 123] use deep learning-based models such as Convolutional Neural Networks (CNN), Long-Short Term Memory Networks (LSTM) to perform the SA task on Telugu data. However, in these works, the corpus is small, used the 2X annotated dataset, and the model's accuracy is low, considering the baseline model.

We did not notice any literature in Telugu language for other NLP tasks. We are the first to handle different NLP tasks in Telugu language and to propose a multi-task GCN architecture for Telugu.

2.4.2 Telugu Pre-trained Embeddings

There have been some concerted and noteworthy efforts to build the pre-trained word-embedding models for the Telugu, including Word2Vec in [87], an n-gram based skip-gram model

to train the FastText embeddings in [13]. Similar to the n-gram approach, sub-word pre-trained word-embeddings based on Byte-Pair Encoding (BPE) are trained on Telugu Wikipedia in [48]. Also, the recent successful multilingual pre-trained Transformer models such as mBERT [34], and Cross-Lingual Language Model (XLM-R) [29], use Telugu corpus as one of the multilingual corpora for training. However, these models do not have high quality representations for all languages. For example, mBERT performs worse than non-BERT models on several downstream tasks for the bottom 30 percent of languages in terms of dataset size upon which mBERT is trained [130]. The main drawback of these methods is that they never tested on large Telugu annotated corpora, and no benchmark results are available for different NLP tasks in Telugu. Recently, [57] introduced a multilingual language model (pre-trained on many Indian languages), IndicBERT, to evaluate the downstream tasks of major eleven Indian languages. However, these models do not focus on specific language and are particularly unwieldy due to their large number of parameters. Therefore, we see it important to explore language models in the monolingual setting for low-resource language, Telugu.

2.5 Differentiater of the Proposed Approaches

From this work, we want to bridge the gap by creating the resources for Telugu. Creating copious data for multiple NLP tasks can overcome the limitations and help the researchers make the best models for Telugu. Hence there is a need to create a corpus, annotate datasets, embeddings, and develop machine learning and deep learning models to understand contextual predictions and decisions better. To our knowledge, no such resources are available for Telugu, and we are the pioneers in creating large corpus, copious labeled Telugu data for four NLP tasks such as sentiment, emotion, hate-speech, sarcasm, clickbait and pre-trained language models.

2.6 Summary

In this chapter, we presented the existing literature on available English feature representations that can be used to build NLP applications. We have presented the text classification NLP tasks in English language. Also, we have presented a literature survey about the NLP task related work done on mono-lingual (Telugu) language.

In the next chapter, we present our initial works that are done with the existing English data. These experiments are used to understand how the feature representations are useful in sentiment and emotion identification tasks.

Chapter 3

Affect in Tweets using Experts Model

In this chapter, we present the work that I have done to understand how the feature representation methods are useful in identifying the intensity of an emotion in emotion identification task.

3.1 Background

Estimating the intensity of emotion has gained significance as modern textual inputs in potential applications like social media, e-retail markets, psychology, advertisements etc., carry a lot of emotions, feelings, expressions along with its meaning. However, the approaches of traditional sentiment analysis primarily focuses on classifying the sentiment in general (positive or negative) or at an aspect level(very positive, low negative, etc.) and cannot exploit the intensity information. Moreover, automatically identifying emotions like anger, fear, joy, sadness, disgust etc., from text introduces challenging scenarios where single tweet may contain multiple emotions with different intensities and some emotions may even co-occur in some of the tweets. In this paper, we propose an architecture, Experts Model, inspired from the standard Mixture of Experts (MoE) model. The key idea here is each expert learns different sets of features from the feature vector which helps in better emotion detection from the tweet. We compared the results of our Experts Model with both baseline results and top five performers of SemEval-2018 Task-1, Affect in Tweets (AIT). The experimental results show that our proposed approach deals with the emotion detection problem and stands at top-5 results.

3.2 Dataset Description

We used the dataset from SemEval-2018 Task 1: AIT [1] for training our system. There is a total of five subtasks: EI-reg (Emotion Intensity regression), EI-oc (Emotion Intensity

[1] `https://competitions.codalab.org/competitions/17751#learn_the_details-datasets`

ordinal classification), V-reg (Valence regression), V-oc (Valence ordinal classification) and E-c (Emotion multi-label classification). Each subtask has three datasets: train, dev, and test. In this paper, we worked on the all five subtasks mentioned above. The dataset details are briefly shown in Table 3.1.

Table 3.1: SemEval-2018 Task-1 Dataset Details

Dataset	train	dev	test	Total
EI-reg, EI-oc				
anger	1701	388	1002	3091
fear	2252	389	986	3011
joy	1616	290	1105	2905
sadness	1533	397	975	2905
V-reg, V-oc	1181	886	3259	2567
E-c	6838	886	3259	10953

3.3 Approach

We took inspiration from the Mixture of Experts (MoE) [51, 91] regression and classification models, where each expert tunes to some set of features out of all the features.

3.3.1 MoE Description

In this subsection, we briefly describe the MoE model to enable the readers to relate our model to MoE architecture. The MoE architecture consists of a number of experts and a gating network. In MoE, there are parameters for each of the expert and a separate set of parameters for gating network. The expert and gate parameters are trained simultaneously using Expectation Maximization [52] or Gradient Descent Approach [53].

Consider the following regression problem. Let $X = \{x^{(n)}\}_{n=1}^{N}$ are N input vectors (samples) and $Y = \{y^{(n)}\}_{n=1}^{N}$ are N targets for each input vector. Then, MoE model is described in terms of parameter $\theta = \{\theta_g, \theta_e\}$ where θ_g is set of the gate parameters and θ_e is sets of the expert parameters. Given a sample x from among N samples, the total probability of predicting target

y can be written in terms of the experts as

$$\begin{aligned} P(y|x,\theta) &= \sum_{i=1}^{I} P(y,x|\theta) \\ &= \sum_{i=1}^{I} P(i|x,\theta_g)P(y|i,x,\theta_e) \\ &= \sum_{i=1}^{I} g_i(x,\theta_g)P(y|i,x,\theta_e) \end{aligned} \tag{3.1}$$

where I is the number of experts, the function $g_i(x,\theta_g) = P(i|x,\theta_g)$ represents the probability of selecting i^{th} expert given x and $P(y|i,x,\theta_e)$ represents the probability of i^{th} expert giving y on seeing x.

The MoE training maximizes the log-likelihood of the probability in equation 3.1 to learn the parameters' set θ [140].

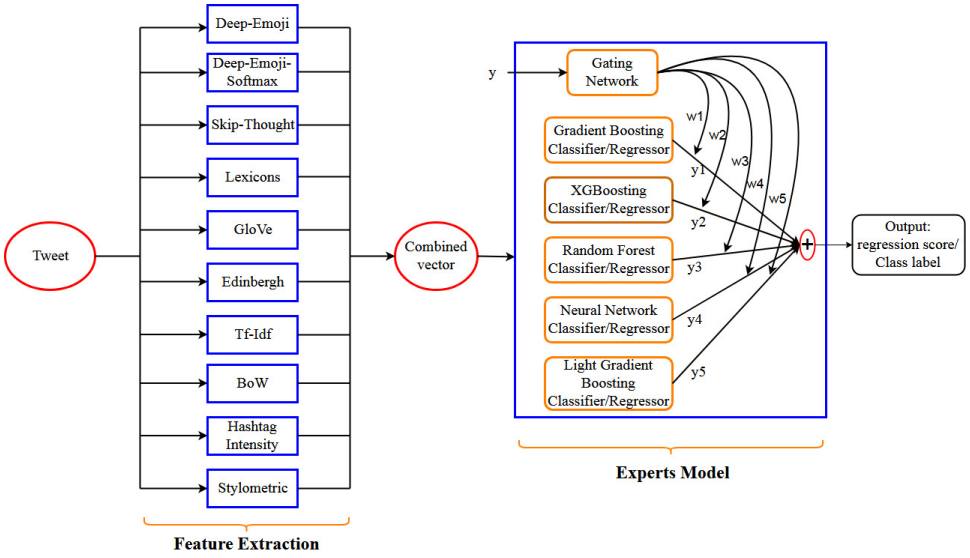


Figure 3.1: Proposed Experts model

3.3.2 Our Proposed Approach

We used the similar architecture, however with some modifications, to measure the intensity of an emotion in a tweet (regression) or predict an emotional intensity (classification). In our proposed approach, we pre-train each expert, to get parameters θ_e, on the training samples, unlike traditional MoE model. Each expert, in itself, can be a separate Regression/Classification model like Multi-level Perceptron (MLP) model or Long Short-Term Memory (LSTM) model

or any other model that best suits the data and task at hand. Once each expert is trained separately, we train only the gating network based on Gradient Descent Approach. The Detailed description of the model is depicted in Figure 3.1 and explained below.

We build different models - Neural Network Classifier/Regressor, Gradient Boosting Classifier/Regressor, XGBoost Classifier/Regressor, Random Forest Classifier/Regressor, Lasso Regressor and Light Gradient Boosting Classifier/Regressor and train each of them with the extracted feature vector of each tweet. We obtain this feature vector by concatenating all of the features discussed in Section 5. We assign parameters θ_g, weights and bias, for each Classifier/Regressor at the gating network. Later, we train the gating network to fit the predicted $\hat{y}$ of each expert i with actual y and learn the best θ_g.

Let $w[i]$ denote weight of each expert i at the gating network. Let $b[i]$ denote the bias term for each expert i at the gating network. Let I be the number of experts. We define the Error function (E) as

$$E = \sum_{i=1}^{I} \frac{1}{2} prob[i] \Big(y[i] - \hat{y}[i] + b[i] \Big)^2$$

where $prob[i]$ is softmax probability of weight $w[i]$, $y[i]$ is the actual y of i^{th} expert for some sample x. Similarly, $\hat{y}[i]$ is the predicted y of i^{th} expert for same sample x. It is to be noted that $\forall i \; y[i] = y$.

For each sample x and y, we train the gating network using the update equations of gradients as follows:

$$\frac{\partial E}{\partial w[i]} = \frac{1}{2} prob[i] \Big(1 - prob[i] \Big) \Big(y[i] - \hat{y}[i] + b[i] \Big)^2$$
$$\frac{\partial E}{\partial b[i]} = prob[i] \Big(y[i] - \hat{y}[i] + b[i] \Big)$$

and

$$w[i] = w[i] - \eta * \frac{\partial E}{\partial w[i]}$$
$$b[i] = b[i] - \eta * \frac{\partial E}{\partial b[i]}$$

where η is the learning rate.

3.4 Preprocessing and Feature Extraction

To preprocess each tweet, first we break all the contractions (like “can’t” to “cannot”, “I’m” to “I am” etc,.) followed by spelling corrections, decoding special words and acronyms (like “e

g" to "eg", "fb" to "facebook" etc,.) and symbol replacements (like "$" to "dollar", "=" to "is equal to" etc). Later, we tokenized each tweet using NLTK tweet tokenizer [2].

The basic idea of using different experts and eclectic features is from the intuition that each expert learn from different aspects of the concatenated features. Hence, we explored and extracted a variety of features; and used only the features which best performs among all the explored ones are explained in the following subsections.

3.4.1 Deep-Emoji Features

Deep-Emoji [41] performs prediction using the model trained on a dataset of 1246 million tweets and achieves state-of-the-art performance within sentiment, emotion and sarcasm detection. We can use the architecture of Deep-Emoji and train the model using millions of tweets from social media to get a better representation of new data. Using the pre-trained Deep-Emoji model, we extracted two different set of features - one, 64-dimensional vector from the softmax layer and the other, 2304-dimensional vector from attention layer.

3.4.2 Word-Embedding Features

In this paper, we tried four different pre-trained word-embedding approaches such as Word2Vec [80], GloVe [95], Edinburgh Twitter Corpus [98] and FastText [13] for generating word vectors. We used the GloVe model of 300 dimensions.

3.4.3 Skip-Thought Features

Skip-Thoughts vectors [67] model is in the framework of encoder-decoder models. Here, an encoder maps words to sentence vector and a decoder is used to generate the surrounding sentences. The main advantage of Skip-Thought vectors is that it can produce highly generic sentence representations from an encoder that share both semantic and syntactic properties of surrounding sentences. Here, we used Skip-Thought vector encoder model to produce a 4800 dimension vector representation of each tweet.

3.4.4 Lexicon Features

We also chose various lexicon features for the model. The lexicon features include AFINN Lexicon [89] (calculates positive and negative sentiment scores from the lexicon), MPQA Lexicon [128] (calculates the number of positive and negative words from the lexicon), Bing Liu Lexicon [11] (calculates the number of positive and negative words from the lexicon), NRC Affect Intensities, NRC-Word-Affect Emotion Lexicon, NRC Hash-tag Sentiment Lexicon, Sentiment140 Lexicon [44] (calculates positive and negative sentiment score provided by the lexicon

[2] `https://www.nltk.org/api/nltk.tokenize.html`

in which tweets are annotated by lexicons), and SentiWordNet [6] (calculates positive, negative, and neutral sentiment score). The final feature vector is the concatenation of all the individual features.

3.4.5 Hash-tag Intensity Features

The work by [81] describes that removal of the emotion word hashtags causes the emotional intensity of the tweet to drop. This indicates that emotion word hashtags are not redundant with the rest of the tweet in terms of the overall intensity. Here, we used Depeche mood dictionary [115] to get the intensities of hashtag words. We average the intensities of all hashtags of a single tweet to get the total intensity score.

3.4.6 Stylometric Features

Tweets and other electronic messages (e-mails, posts, etc.) are written far shorter, way more informal and much richer in terms of expressive elements like emoticons and aspects at both syntax and structure level, etc. Common techniques use stylometric features [4] which are categorized into 5 different types: lexical, syntactic, structural, content specific, and idiosyncratic. In this paper, we used 7 stylometric features such as "number of emoticons", "number of nouns", "number of adverbs", "number of adjectives", "number of punctuations", "number of words", and "average word length".

3.4.7 Unsupervised Sentiment Neuron Features

Unsupervised sentiment neuron model [103] provides an excellent learning representation of sentiment, despite being trained only on the text of Amazon reviews. A linear model using this representation results in good accuracy. This model represents a 4096 feature vector for any given input tweet or text.

3.5 Experimental Setup and Results

To train our proposed approach, we consider a total of five learning models, one for each expert: Gradient Boosting, XGBoost, Light Gradient Boosting, Random Forest, and Neural Network(NN) for subtasks EI-reg and V-reg. While for the subtasks EI-oc and V-oc, we consider all the models except NN model. For subtask E-c, we consider all the models except Light Gradient Boosting model.

Table 3.2: Model-Parameters

Model	Parameters
Gradient Boosting	n_estimators: 3000, Learning rate: 0.05 Max_depth: 4
XGBoosting	n_estimators: 100 learning_rate: 0.1 max_depth: 3
Neural Network	Optimizer: adam Activation : relu
Random Forest	n_estimators: 250 max_depth: 4
Light Gradient Boosting	n_estimators: 720 learning_rate: 0.05 num_leaves: 5

Table 3.3: Comparison of Regression results of various models with our Experts Model

	EI-reg (Pearson (all instances))					EI-reg (Pearson (gold in 0.5-1))				
Team	macro-avg	anger	fear	joy	sadness	macro-avg	anger	fear	joy	sadness
SeerNet	0.799(1)	0.827	0.779	0.792	0.798	0.638(1)	0.708	0.608	0.708	0.608
NTUA-SLP	0.776(2)	0.782	0.758	0.771	0.792	0.610(2)	0.636	0.595	0.636	0.595
PlusEmo2Vec	0.766(3)	0.811	0.728	0.773	0.753	0.579(5)	0.663	0.497	0.663	0.497
psyML	0.765(4)	0.788	0.748	0.761	0.761	0.593(4)	0.657	0.541	0.657	0.541
Experts Model	**0.753(5)**	**0.789**	**0.742**	**0.748**	**0.733**	**0.598(3)**	**0.656**	**0.582**	**0.546**	**0.608**
Median Team	0.653(23)	0.654	0.672	0.648	0.635	0.490(23)	0.526	0.497	0.420	0.517
Baseline	0.520(37)	0.526	0.525	0.575	0.453	0.396(37)	0.455	0.302	0.476	0.350

Note : The numbers inside parenthesis in both macro-avg columns represent the rank

3.5.1 Training Strategy

At the input layer, we used a concatenation vector of all features: Deep-Emoji, Skip-Thought, Lexicons, Stylometric, BoW, Tf-IDF, Glove, Word2Vec, Edinburgh, and HashTagIntensity which is same for each expert. We combined both training and dev data and used them for training our model. The training model is validated by stratified K-fold approach in which the model is repeatedly trained on K-1 folds and the remaining one fold is used for validation.

In order to tune the hyper-parameters of our experts model, we adopt a grid search cross-validation for each learning model. Using grid search cross-validation, we set the various types of parameters based on the learning model. Table 3.2 shows the parameter settings for all experts.

Table 3.4: Comparison of Classification results of various models with our Experts Model

Team	EI-oc (Pearson (all classes))					EI-oc (Pearson (some emotion))				
	macro-avg	anger	fear	joy	sadness	macro-avg	anger	fear	joy	sadness
SeerNet	0.695(1)	0.706	0.637	0.720	0.717	0.547(1)	0.559	0.458	0.610	0.560
PlusEmo2Vec	0.659(2)	0.704	0.528	0.720	0.683	0.501(4)	0.548	0.320	0.604	0.533
psyML	0.653(3)	0.670	0.588	0.686	0.667	0.505(3)	0.517	0.468	0.570	0.463
Amobee	0.646(4)	0.667	0.536	0.705	0.673	0.480(5)	0.458	0.367	0.603	0.493
Experts Model	**0.636(5)**	**0.658**	**0.576**	**0.666**	**0.644**	**0.520(2)**	**0.493**	**0.502**	**0.579**	**0.509**
Median Team	0.530(17)	0.530	0.470	0.552	0.567	0.415(17)	0.408	0.310	0.494	0.448
Baseline	0.394(26)	0.382	0.355	0.469	0.370	0.296(26)	0.315	0.183	0.396	0.289

Note : The numbers inside parenthesis in both macro-avg columns represent the rank

Table 3.5: Comparison of Valence-reg results of various models with our Experts Model

Team	V-reg (Pearson)	
	(all instances)	(gold in 0.5-1)
SeerNet	0.873	0.697
TCS Research	0.861	0.680
PlusEmo2Vec	0.860	0.691
NTUA-SLP	0.851	0.688
Amobee	0.843	0.644
Experts Model	**0.830**	**0.670**
Median Team	0.784	0.509
Baseline	0.585	0.449

Table 3.6: Comparison of Valence-oc results of various models with our Experts Model

Team	V-oc (Pearson)	
	(all instances)	(gold in 0.5-1)
SeerNet	0.836	0.884
PlusEmo2Vec	0.833	0.878
Amobee	0.813	0.865
psyML	0.802	0.869
EiTAKA	0.796	0.838
Experts Model	**0.738**	**0.773**
Median Team	0.682	0.754
Baseline	0.509	0.560

3.5.2 Results

To evaluate our computational model, we compare our results with SemEval-2018 Task-1 (Affect in Tweets) baseline results, top five performers and Median Team (as per SemEval-2018 results). The results in the EI-reg, EI-oc, V-reg, V-oc, E-c are shown in Tables 3.3, 3.4, 3.5, 3.6, 3.7 respectively. The tables illustrate (a) the results obtained by our proposed approach, (b) top five performers in SemEval-2018, (c) the results obtained by a baseline SVM system using unigrams as features and (d) Median Team among all submissions. From the Tables 3.3 and 3.4, we observe that our model (considering only macro-average for Pearson Correlation) for EI-reg and EI-oc stands within 5 places among 48 submissions. A quick walk-through of Table 3.3 for individual emotions shows that anger and fear ranks 3^{rd} and 4^{th} respectively for EI-reg Pearson(all instances) and for EI-reg Pearson(gold in 0.5-1), fear stands at 3^{rd} position and sadness equals score of top performer. Similarly, Table 3.4 for classification results shows that anger and fear ranks 4^{th} and 3^{rd} respectively for EI-oc Pearson(all classes) and for EI-oc Pearson(some emotion), anger, fear, joy and sadness stands at positions 4^{th}, 1^{st}, 4^{th} and 3^{rd} respectively. It is to be noted that in both tables 3.3 and 3.4, the numbers inside parenthesis under column "macro-avg" represent the rank according to macro-avg Pearson scores. These

Table 3.7: Comparison of E-c results of various models with our Experts Model

Team	E-c (acc.)	(micro F1)	(macro F1)
NTUA-SLP	0.588(1)	0.701	0.528
TCS Research	0.582(2)	0.693	0.530
PlusEmo2Vec	0.576(4)	0.692	0.497
psyML	0.574(5)	0.697	0.574
Experts Model	**0.578(3)**	**0.691**	**0.581**
Median Team	0.471(17)	0.599	0.464
Baseline	0.442(21)	0.570	0.443

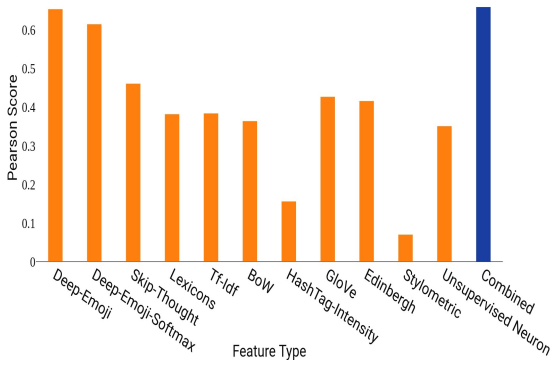
Note : The numbers inside parenthesis in accuracy column represent the rank

values shows that our model stands at 3^{rd} and 2^{nd} positions in EI-reg Pearson(gold in 0.5-1) and EI-oc Pearson(some emotion) respectively. Tables 3.5 and 3.6 illustrate that the results from our model are among the top 10 submissions of subtasks V-reg and V-oc. Table 3.7 shows the results of multi-label emotion classification (11 classes). Our model is among the top 3 submissions for Jaccard similarity (accuracy) metric, in top 5 for micro F1 metric and topped the submissions for macro F1 metric.

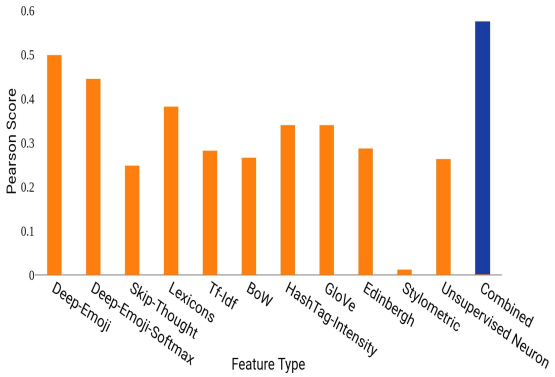
3.5.3 Metrics

We use the competition metric, Pearson Correlation Coefficient with the Gold ratings/labels from SemEval-2018 task-1 AIT for EI-reg, EI-oc, V-reg and V-oc. Further, macro-average was calculated by averaging the correlation scores of four emotions: anger, fear, joy, and sadness for the tasks EI-reg and EI-oc. Along with Pearson Correlation Coefficient, we use some additional metrics for each subtask. The additional metric used for EI-reg and V-reg tasks was to calculate the Pearson correlation only for a subset of test samples where the intensity score was greater than or equal to 0.5. For the classification subtasks EI-oc and V-oc, we use the additional metric Pearson correlation calculated only for some emotion like low emotion, moderate emotion, or high emotion. However, for the multi label emotion classification E-c, we used the official evaluation metrics Jaccard Similarity (accuracy), micro average F1 score and macro average F1 score of all the classes.

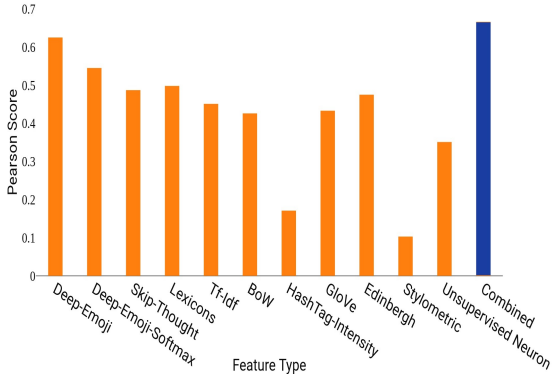
Figure 3.2 shows the influence of each feature type on scores for predicting the intensity or emotion. We can observe from Figure 3.2 that for “Deep-Emoji” and “Deep-Emoji-Softmax” features, Pearson scores are dominating other feature types. Feature types - Skip-Thought, Lexicons, Glove, and Edinburgh features are contributing approximately similar in each of the four emotions. However, Stylometric features and features from Unsupervised sentiment neurons are performing worse.



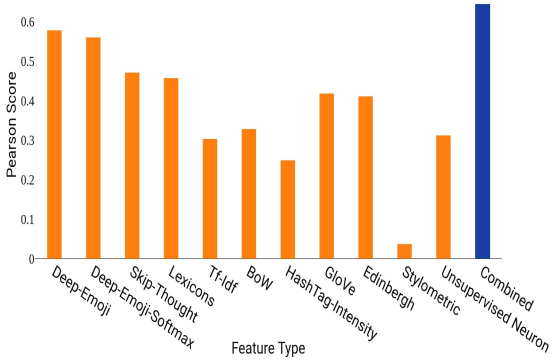
(a) Emotion: Anger



(b) Emotion: Fear



(c) Emotion: Joy



(d) Emotion: Sadness

Figure 3.2: Feature Importance: Comparison of Pearson Scores for each feature vector and concatenated vector

3.6 Conclusion

In this paper, we have proposed a novel approach inspired from standard Mixture-of-Experts model to predict the intensity of an emotion(Regression) or level of an emotion (Classification) or multi-label emotion classification. Experiment results show that our proposed approach can effectively deal the emotion detection problem and stands at top-5 when compare with SemEval-2018 Task-1 AIT results and baseline results. The source code is publicly available at `https://goo.gl/NktJhF` so that researchers and developers can work on this exciting problem collectively.

In the next chapter, we present our second work on English language where we evaluated different feature representations using experts model on sentiment analysis task.

Chapter 4

Evaluating Word Representations with Sentiment Analysis

In this chapter, we evaluated the combination of different feature representations using mixture of experts for sentiment analysis task.

4.1 Background

Neural word embeddings have been able to deliver impressive results in many Natural Language Processing tasks. The quality of the word embedding determines the performance of a supervised model. However, choosing the right set of word embeddings for a given dataset is a major challenging task for enhancing the results. In this paper, we have evaluated neural word embeddings on sentiment analysis task in two steps: (i) proposed a mixture of classification experts (MoCE) model for sentiment classification task, (ii) to compare and improve the classification accuracy by different combination of word embedding as first level of features and pass it to cascade model inspired by gcForest for extracting diverse features. We argue that in the first step, each expert learns a certain positive or negative examples corresponding to its category and in the second step resulting features on a given task (polarity identification) can achieve competitive performance with state-of-the-art methods in terms of accuracy, precision and recall using gcForest.

4.2 Model Architecture

We use a mixture of experts based model, whose architecture is inspired from [53]. The mixture of experts architecture is composed of gating network and several expert networks, each of which solves a function approximation problem over a local region of the input space. The detailed overview of our model is shown in Figure 4.1 where the input is a text vector extracted from recently successful neural embeddings such as Word2Vec, GloVe, ELMo, & BERT. These input features pass through both the gating network and two of the experts.

The gating network uses a probabilistic model to choose the best expert for a given input text vector.

4.2.1 MoCE Architecture

Given an input feature vector $\mathbf{x}$ from the one of the neural word embedding method, we model its posterior probabilities as a mixture of posteriors produced by each expert model trained on $\mathbf{x}$.

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \sum_{j=1}^{K} P(S_j|\mathbf{x}, \theta_0) p(\mathbf{y}|\mathbf{x}, S_{\theta_j}) \\ &= \sum_{j=1}^{K} g_{S_j}(\mathbf{x}, \theta_0) p(\mathbf{y}|\mathbf{x}, S_{\theta_j}) \end{aligned} \tag{4.1}$$

Here, $P(S_j|\mathbf{x}, \theta_0) = g_{S_j}(\mathbf{x}, \theta_0)$ is the probability of choosing S_j^{th} expert for given input $\mathbf{x}$. Note that $\sum_{j=1}^{K} g_{S_j}(\mathbf{x}, \theta_0) = 1$ and $g_{S_j}(\mathbf{x}, \theta_0) \geq 0,\ \forall j \in [K]$. $g_{S_j}(\mathbf{x}, \theta_0)$ is also called gating function and is parameterized by θ_0.

Since the class labels $\{\mathbf{y}_1, \mathbf{y}_2,...,\mathbf{y}_n\}$ are independent and identically distributed sample of outcome variables from a population modelled by a K-component finite mixture model. Here, the outcome variable is discrete (either positive or negative sentiment). Due to this reason, in this paper, we choose $p(\mathbf{y}|\mathbf{x}, S_{\theta_j})$ as a Gaussian probability density for each of the experts, denoted by:

$$p(\mathbf{y}|\mathbf{x}, S_{\theta_j}) = \frac{1}{(|\sigma_j|2\pi)^{1/2}} \exp\left(-\frac{1}{2\sigma_j^2}(\mathbf{y} - W_j\mathbf{x})^T(\mathbf{y} - W_j\mathbf{x})\right) \tag{4.2}$$

where $S_{\theta_j} \in \mathbb{R}^{m \times n}$ is the weight matrix associated with the S_j^{th} expert. Thus, $S_{\theta_j} = \{W_j\}$. We use softmax function for the gating variable $g_{S_j}(\mathbf{x}, \theta_0)$.

$$g_{S_j}(\mathbf{x}, \theta_0) = \frac{\exp\left(\mathbf{v}_j^T\mathbf{x}\right)}{\sum_{i=1}^{K} \exp\left(\mathbf{v}_i^T\mathbf{x}\right)} \tag{4.3}$$

where $\mathbf{v}_j \in \mathbb{R}^n,\ \forall j \in [K]$. Thus, $\theta_0 = \{\mathbf{v}_1, \ldots, \mathbf{v}_K\}$. Let Θ be the set of all the parameters involved for the K-experts. Thus, $\Theta = \{\theta_0, (W_1), \ldots, (W_K)\}$. Here, we train the MoCE model and update the weights iteratively using expectation-maximization (EM) algorithm.

4.2.2 Multigrained gcForest Architecture

In order to improve the classification performance of each dataset, we passed the input feature vector to a multigrain gcForest model for better feature representation. The gcForest

Table 4.1: Model Parameters of Cascading gcForest

Model	Parameters
XGB	n_foldss: 5 n_estimators: 100 max_depth: 5 learning_rate: 0.1
LGBM	n_foldss: 5 n_estimators: 100 max_depth: 5 learning_rate: 0.1
RF	n_foldss: 5 n_estimators: 100
ET	n_foldss: 5 n_estimators: 100

Table 4.2: Comparison of word embedding results of 20 domains of Dranziera dataset with our MoCE Model. The values in the table indicates the percentage of positive reviews captured by Expert1 and percentage of negative reviews captured by Expert2.

Domain	Word2vec		GloVe		BERT		ELMo	
	Expert1	Expert2	Expert1	Expert2	Expert1	Expert2	Expert1	Expert2
Amazon_Instant_Video	0.81	0.86	0.81	0.86	0.54	0.55	0.71	0.72
Automotive	0.81	0.85	0.85	0.82	0.54	0.55	0.72	0.72
Baby	0.73	0.87	0.97	0.05	0.61	0.67	0.77	0.72
Beauty	0.02	0.98	0.86	0.82	0.55	0.54	0.68	0.71
Books	0.82	0.83	0.84	0.83	0.57	0.57	0.75	0.68
Clothing_Accessories	0.90	0.79	0.85	0.88	0.66	0.74	0.78	0.73
Electronics	0.98	0.04	0.85	0.81	0.56	0.55	0.73	0.75
Health	0.80	0.83	0.81	0.84	0.59	0.55	0.71	0.73
Home_Kitchen	0.81	0.87	0.88	0.83	0.59	0.59	0.69	0.73
Movies_TV	0.85	0.80	0.03	0.97	0.54	0.57	0.72	0.76
Music	0.80	0.86	0.85	0.80	0.64	0.62	0.78	0.79
Office_Products	0.99	0.02	0.87	0.80	0.65	0.64	0.80	0.82
Patio	0.03	0.99	0.99	0.04	0.31	0.55	0.69	0.67
Pet_Supplies	0.82	0.80	0.82	0.80	0.54	0.56	0.71	0.73
Shoes	0.92	0.84	0.92	0.86	0.60	0.65	0.77	0.75
Software	0.82	0.84	0.87	0.71	0.55	0.55	0.71	0.73
Sports_Outdoors	0.78	0.87	0.79	0.87	0.58	0.59	0.69	0.73
Tools_Home_Improvement	0.85	0.78	0.85	0.79	0.55	0.54	0.70	0.77
Toys_Games	0.88	0.85	0.87	0.85	0.45	0.43	0.75	0.73
Video_Games	0.81	0.83	0.04	0.99	0.43	0.39	0.71	0.73

model we motivate from [149], where the cascade structure, as illustrated in Figure 4.2, where each cascading level receives input from the preceding level and the processed result passed to the next level.

The raw input feature vector is given to gcForest with different dimension associated with pretrained embeddings. Each cascading level contains different ensemble based forest models i.e an ensemble of ensembles yields the diversity in feature construction. Here, each forest produces a class distribution for each instance and finally estimate the average of all class distributions across the ensemble based forests gives an output vector. The output vector is concatenated with the original feature vector and passed to the next cascading level. In order to avoid the risk of over fitting, each forest uses K-fold cross-validation to produce the class vector. Moreover,

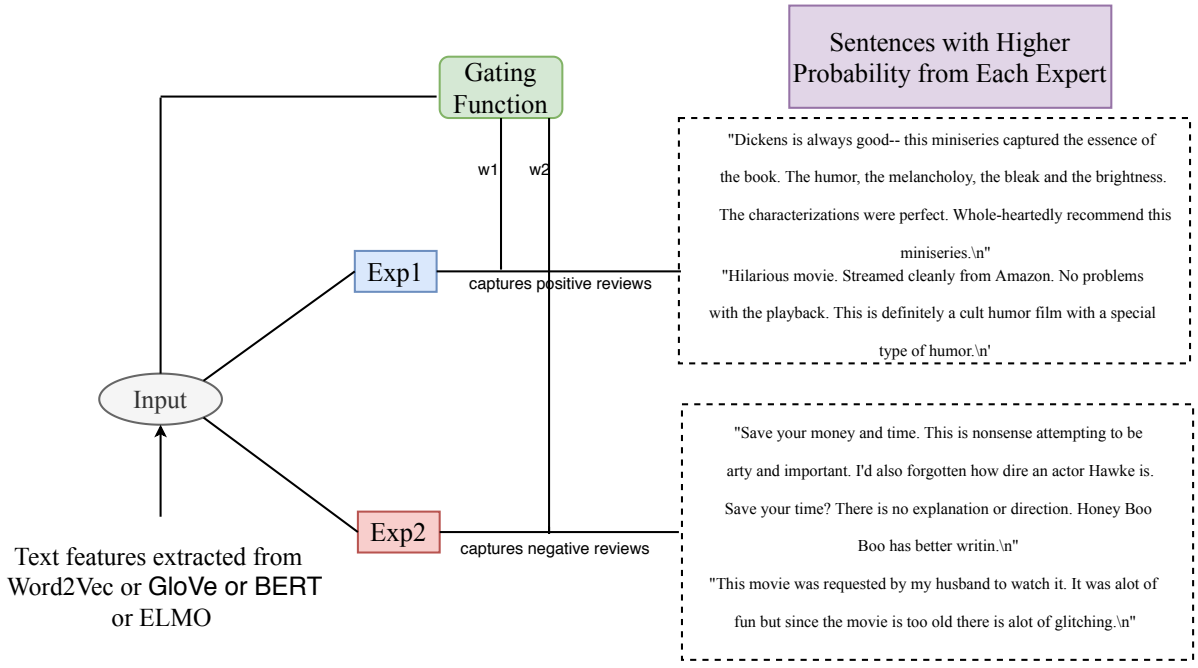


Figure 4.1: Proposed Mixture of Classification Experts (MoCE) model. Here, Expert1 captures positive reviews and Expert2 captures negative reviews.

the complexity of a model can be controlled by checking the training error and validation error to terminate the process when the training is adequate.

4.3 Experimental Setup

In order to evaluate the word embeddings, we choose sentiment analysis task to perform the experiments. Here, we briefly describe the dataset Amazon Product Reviews.

4.3.1 Dataset Description

Amazon product domains: This corpus is a collection of 20 product reviews derived from Task-1 of ESWC Semantic Challenge-2019. The 20 different Amazon product domains names are mentioned here [1], and this corpus belongs to sentiment analysis task. The data for the Task-1 will consist of 50k reviews for each domain of which 25k reviews are positive and 25k reviews are negative. The evaluation metrics for method evaluation are precision, recall, and macro F1-score.

[1] `http://www.maurodragoni.com/research/opinionmining/events/challenge-2019/`

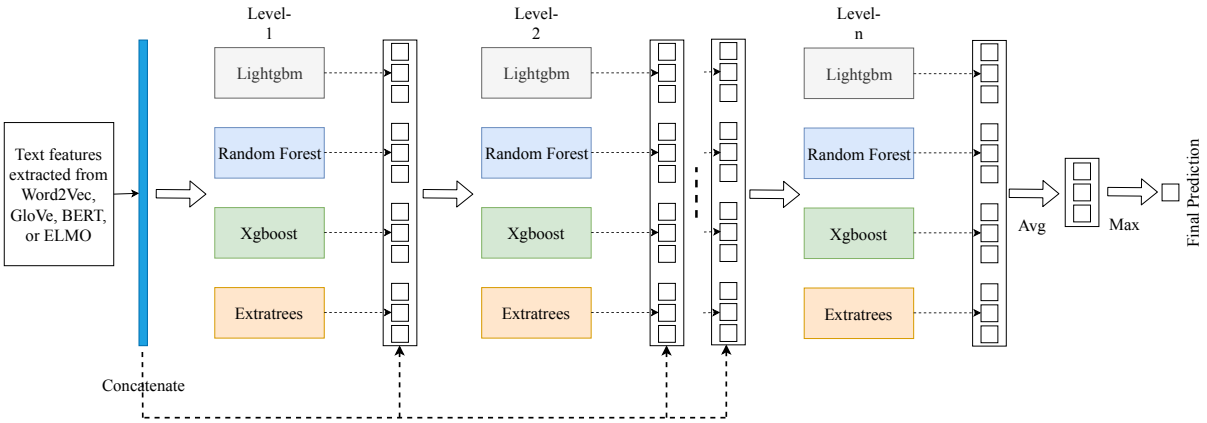


Figure 4.2: Cascading gcForest Architecture

4.3.2 Feature Extraction

In this paper, we mainly focused on four successful pretrained word embeddings such as: Word2Vec (embeddings are of 300 dimensions) [80], GloVe (embeddings are of 300 dimensions) [95], BERT (embeddings are 768 dimensions each) [33], and ELMo (embeddings are 1024 dimensions each) [97].

4.3.3 Training Strategy

Using the approach discussed in Section 2, we trained a separate mixture of classification experts model (MoCE) for the dataset Amazon Product Reviews with the associated task sentiment analysis using all the embeddings. The input to the MoCE model is a text vector and output is the corresponding classes based on a specific task. Here, we select the number of experts based on the number of output classes. The gating function selects one of the experts with higher probability score for the corresponding input. The selected expert predicts the target label using that particular expert weights. Both expert parameters and gating parameters are updated using the iterative expectation-maximization (EM) algorithm. The training model is validated by K-fold approach in which the model is repeatedly trained on K-1 folds and the remaining one fold is used for validation. The proposed model is trained until the model reaches the convergence with a lower bound of $1e^{-5}$ or a maximum of 100 iterations.

4.4 Results and Discussion

Here, we conducted the experiments in two steps. In the first step, we evaluated the four word embeddings using MoCE model and the second step describes better feature representation

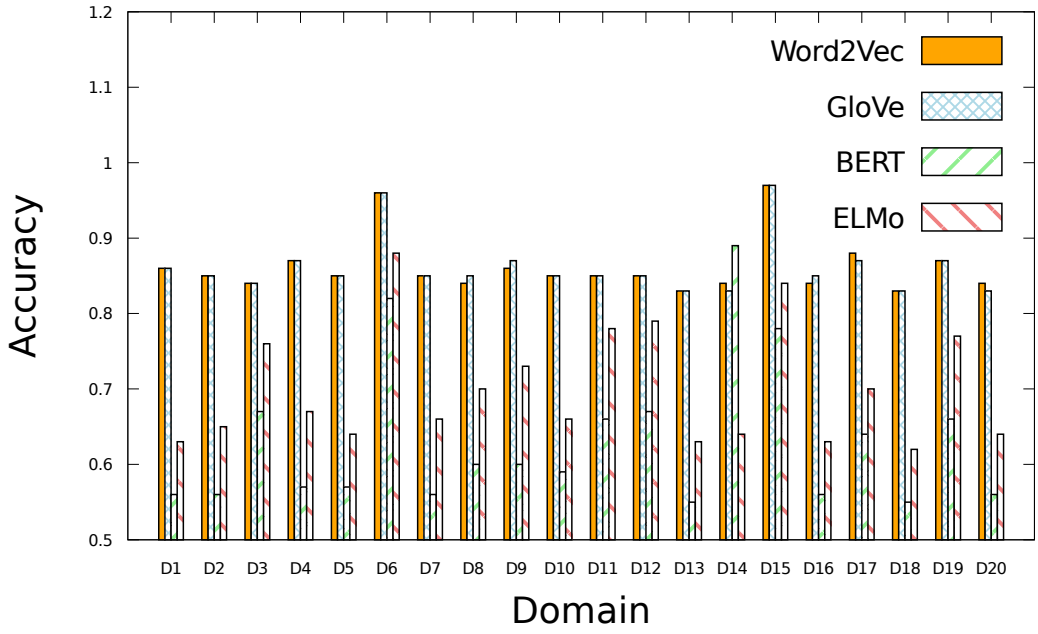


Figure 4.3: Figure presents the accuracy of amazon 20 products using gcForest on four word embeddings Word2Vec, GloVe, BERT, and ELMo.

using cascading gcForest outperforms the state-of-the-art results on amazon product review datasets.

4.4.1 Evaluation of Embeddings using MoCE

Experiments are conducted on the 20 Amazon product domains dataset by passing input as text vector extracted from recent successful neural word embeddings and output as corresponding target classes positive or negative. We split the dataset into 40000 reviews in training and 10000 reviews into testing. The MoCE model performance was evaluated by training and testing the different subsets of the 50000 reviews in a 5-fold cross-validation scheme.

Table 4.2 presents the performance results of each embedding scheme where the two experts discriminate both positive and negative examples. From the table 4.2, we can observe that both GloVe and Word2Vec embeddings having better discrimination where one of the experts captures majority positive sentiment examples as other expert capture more negative sentiment examples. Here, we use test dataset of total 10000 examples out of which 5000 samples are positive and 5000 samples are negative. For example, from the Table 4.2 consider the Shoes domain dataset, for the GloVe Embedding: expert1 captures 92% positive sentiment samples and expert2 captures 86% negative sentiment samples, shows better discrimination and similarly with the Word2Vec and ELMo. Word embeddings like Word2Vec and GloVe embedding feature

as input, MoCE model isolate the positive and negative examples by two experts. In contrary, for the domains Baby, Electronics, Office_Products and Patio (here expert1 only captures all the positive and negative samples), this is mainly because of expressing the opinion in reviews are almost similar in both classes. However, in the case of BERT and ELMo embedding: both experts isolate the samples for all the domains to capturing of context-sensitive information.

4.4.2 Polarity Identification using gcForest

Using the MocE results described in Table 4.2, we can observe the better feature representation of each pretrained word embedding model based on the experts which discriminate the positive or negative samples. In order to validate and improve the classification performance, we also built the cascading gcForest classification model described in section 4.2.2. We use four ensemble forest models such as LightGBM [59], XGboost [20], Random Forest, and Extra Trees classifier in each cascading layer. The configuration of the gcForest model is shown in Table 4.1. Here, we use a 5-fold cross-validation method to avoid the overfitting problem. With this method, the model outperforms the state-of-the-art results mentioned in [37] for different combination features such as GloVE, Word2Vec, ELMo & BERT as shown in Table 4.3. We also improve the classification performance of each domain dataset by using the above mentioned four embeddings. Since, gcForest doesnot require more hyper-parameters and deeper layers to train to achieve good performance and very fast to train.

Figure 4.3 illustrates each domain results for all the pretrained embeddings with an evaluation metric accuracy. From the figure 4.3, we can observe that Word2Vec, GloVe, and ELMO methods perform better when compared to BERT embeddings in terms of accuracy and similar comparison we observed in table 4.2 using an evaluation metric F1-score. One of the main reason why BERT & ELMo do not perform better than Word2Vec & GloVe is that to fine-tune language models (LMs) likes BERT/ELMo for a specific dataset training for few epochs getting better results instead of simply using pretrained embeddings. In Table 4.3, we describes the comparison between previous state-of-the-art methods and using gcForest. The combination of word embedding results comparison we observed in Table 4.3 and it outperforms the state-of-the-art results.

4.5 Conclusion

Neural word embeddings have been able to deliver impressive results in many Natural Language Processing tasks. However, choosing the right set of word embeddings for a given dataset is a major challenging task for enhancing the results. In this paper, we have evaluated four neural word embedding methods such as Word2Vec, GloVe, ELMo, & BERT on sentiment analysis task in two steps (i) a mixture of classification experts (MoCE) model for sentiment

Table 4.3: Detailed results of domains (Dom) of Amazon product reviews dataset by the Baselines, existing method results and by passing combination of word embeddings to gcForest

	Tested System (Macro F1- Score)							
Dom	SVM	ME	DBP	DDP	CNN	GWE	NS	gcF
(1)	0.70	0.70	0.72	0.71	0.80	0.80	0.80	**0.87**
(2)	0.72	0.71	0.72	0.70	0.73	0.79	0.85	**0.87**
(3)	0.69	0.72	0.71	0.69	0.84	0.79	0.85	**0.86**
(4)	0.69	0.72	0.74	0.73	0.82	0.81	0.85	**0.88**
(5)	0.69	0.69	0.69	0.69	0.78	0.75	0.79	**0.86**
(6)	0.69	0.72	0.80	0.78	0.77	0.81	0.86	**0.97**
(7)	0.68	0.69	0.73	0.70	0.79	0.77	0.86	**0.87**
(8)	0.67	0.66	0.69	0.69	0.78	0.79	0.86	**0.86**
(9)	0.72	0.69	0.71	0.69	0.75	0.82	0.87	**0.88**
(10)	0.73	0.72	0.70	0.71	0.75	0.79	0.80	**0.86**
(11)	0.69	0.65	0.71	0.72	0.76	0.77	0.80	**0.86**
(12)	0.73	0.73	0.72	0.70	0.79	0.80	0.87	**0.87**
(13)	0.69	0.71	0.70	0.69	0.86	0.80	0.86	**0.86**
(14)	0.68	0.73	0.67	0.66	0.82	0.79	0.84	**0.85**
(15)	0.67	0.73	0.83	0.81	0.81	0.84	0.86	**0.97**
(16)	0.74	0.69	0.72	0.71	0.79	0.76	0.85	**0.86**
(17)	0.67	0.73	0.71	0.71	0.76	0.81	0.87	**0.89**
(18)	0.73	0.73	0.68	0.69	0.79	0.79	0.85	**0.85**
(19)	0.66	0.69	0.74	0.71	0.77	0.84	0.86	**0.88**
(20)	0.69	0.70	0.70	0.70	0.72	0.78	0.82	**0.84**

SVM (Support Vector Machines), ME (Maximum Entropy)
DBP (Domain Belonging Polarity), NS (NeuroSent)
DDP (Domain Detection Polarity), gcF(gcForest)
CNN (Convolutional Neural Networks)
GWE(Google Word Embeddings)

classification task, (ii) to compare and improve the classification accuracy by different combination of word embedding as first level of features and pass it to cascade model inspired by gcForest for extracting diverse features. In the future, we plan to experiment on all NLP tasks by using a hierarchical mixture of experts and conduct experiments on other standard datasets with a primary focus on all aspects of word embeddings.

From the further parts of the thesis we are presenting how we solved popular NLP tasks in Telugu language. We considered five NLP tasks for this purpose. In the next chapter, we present the general annotation schema for the five NLP tasks in Telugu language.

Chapter 5

General Annotation Schema for Five NLP Tasks in Telugu

In this chapter, we will explain the annotation process, annotation guidelines, the tool we created for annotation, and the details regarding the compensation to the annotators.

5.1 Telugu Raw Corpus Creation

The absence of large corpus in Telugu necessitated the construction of corpus to Telugu data. Sentences are collected from Telugu websites like visalandhra [1], telugutimes [2], telugumirchi [3], telugu360 [4], surya [5], prajasakti[6], NTV [7], namastheandhra [8], greatandhra [9], eenadu [10], andhrabhoomi [11]. To further broaden the scope of our data, we selected the sentences from different domains such as 'Movies', 'Sports', and 'Political'. Our Telugu corpus consists of 63,78,180 sentences crawled from the above-mentioned websites. Besides, we added the available Wikipedia dump of 16,37,408 sentences to our corpus. The total corpus has 80,15,588 sentences.

5.2 Data Annotation process

For any NLP text classification annotation process, the main task is identifying the main contextual words that lead to the correct label (for SA, is it "positive", "negative" or "neutral"). So once identified, they can be used as information for further training the machine learning and deep learning models. The annotation process involves identifying the main contextual

[1] https://visalaandhra.com/
[2] http://www.telugutimes.net
[3] http://www.telugumirchi.com
[4] https://www.telugu360.com
[5] www.telugu.suryaa.com
[6] http://www.prajasakti.com
[7] https://www.ntvtelugu.com
[8] http://www.namasteandhra.com
[9] https://telugu.greatandhra.com
[10] https://www.eenadu.net
[11] http://www.andhrabhoomi.net

words in raw text and categorizing them into various predefined labels. These labels can be chosen based on the requirement or the granularity to which we want to identify sentiment or emotion or hate speech, and so on. Some of the most common labels for SA tasks are "positive", "negative", and "neutral". The basic labels for EI task are "angry", "fear", "sad", and "happy". The HS, SAR, and clickbait labels are "yes" and "no". Some of the most common labels for the NER task include names, place, organizations, temporal expressions, medical terms, quantities, monetary values, etc.

The annotation process can be divided into two sub-problems. The first one is to identify the meaningful words from a sentence that contains the proper information to decide the label for that sentence. The second is to assign a fitting label for that sentence from the predefined list of labels for a specific task.

We select five users (5X) to annotate the data, all at the primary level of linguistic background and fluent Telugu native speakers. Each user has explained the annotation process by reading the guidelines and examples. All the annotators should read the sentence carefully and determine whether or not it expresses the sentiment, emotion, hate speech, sarcasm, or clickbait. If so, then categorize each task into the respective labels given in the guidelines. The annotator had to correctly annotate at least 80% sentences for four tasks to qualify and complete the annotation process.

We randomly chose 100 sentences from the data at the initial step and asked each user to provide the labels for different NLP tasks. After the annotation, we manually checked the corresponding labels and discussed them with each user for justification. Further, we give the correct labels for wrongly annotated examples and explain the reason for choosing that label. We iterate the above steps three times to ensure that the quality of annotations should be at least 80%. Users who successfully passed the initial set of sentences annotated the remaining sentences of the dataset. Of those sentences, 15% were additional hidden test sentences to continuously assess the five annotator's quality. Accuracy of 80% on test sentences was the threshold for including their annotations in the final dataset. To avoid the redundant text, we considered the Jaccard similarity for discarding duplicate sentences (above 90% match) and reported the datasets.

5.3 Annotation Guidelines for Five Tasks in Telugu

We followed different guidelines for each NLP task to annotate the data. To assign a proper label, we need to look into the word, and sometimes it can be even a group of words like chunks or phrases. For example, in the sentence "I do not like this movie", we need to consider two words "not" and "like" to assign the sentiment label like "negative", emotion labeled as "sad", hate speech, sarcasm, and clickbait label as "no". The sentences will be tagged with the following labels based on the current text classification annotation task's specific NLP task.

5.3.1 Telugu Sentiment Analysis (SA):

The five annotators has to read the Telugu sentence and identify whether the sentence has "positive", "negative", or "neutral" sentiment. Then each annotator has to select the fitting label to that sentence by reading the guidelines provided to them. The sample of annotated sentences in Telugu, WX, and English format for the SA is given in Figure 5.1.

Sentence	Label
సరికొత్త కథా, కథనాలతో తెరకెక్కిన ఈ సినిమా ప్రేక్షకుడికి ఓ సరి కొత్త అనుభూతిని ఇస్తుంది. sarikoVwwa kaWA, kaWanAlawo weVrakeVkkina I sinimA prekRakudiki o sarikoVwwa anuBUwini iswuMxi. This movie that hit the screens with a novel story and screenplay will give a novel experience to the audience.	Positive
రేవేన్యూ సిబ్బంది నిర్లక్ష్యంపై రైతులు ఆగ్రహించారు. reVveVnyU sibbaMxi nirlakRyaMpE rEwulu AgrahiMcAru. The farmers were outraged over the negligence of the revenue staff.	Negative
లోక్ సభ సమావేశాలు రెండో రోజు ప్రారంభం కాగా, స్పీకర్ ఎన్నిక జరగనుంది. lok saBa samAveSAlu reVMdo roju prAraMBaM kAgA, spIkar eVnnika jaraganuMxi. The Speaker's election is scheduled for the second day of Lok Sabha meetings.	Neutral

Figure 5.1: Sample of annotated sentences for SA task in Telugu, WX and English

5.3.2 Telugu Emotion Identification (EI):

For EI task, the annotators should read the telugu sentence and identify the fitting label. The labels we considered for Telugu EI are "angry", "fear", "happy", "sad", or "no emotion". The annotators were required to select one of the five labels for each Telugu sentence. To the best of our knowledge, we are the first that leverage the annotations and consider the task of EI for the Telugu language. Figure 5.2 shows the sample of emotion sentences along with labels reported in Telugu, WX, and English format.

5.3.3 Telugu Hate-speech Detection (HS):

The content on social media platforms like Twitter, YouTube, and Facebook is harmful; the automatic removal of hate-speech content prevents the spreading of hate-speech information. Many researchers have been working for the past decade for automated hate-speech detection in the English [68, 145, 127, 90]. However, we need a Telugu system that automatically identifies hate-speech comments where users post comments in their native language. To the best of our knowledge, we are the first that leverage the annotations and consider the task of hate-speech for the Telugu language. Although hate-speech detection is a challenging problem due to its content difference between users, creating a labeled data and automated system describes the variance difficulties that arise from the text. For the hate-speech task, annotators were required to select a label for hate speech. The labels for hate speech are "yes" and "no". The examples in Telugu, WX, and English format for hate speech are given in Figure 5.3.

Sentence	Label
వరద వచ్చే నాటికి పనులు ఎందుకు పూర్తి కాలేదని జగన్ ఆగ్రహం వ్యక్తం చేశారు. varaxa vacce nAtiki panulu eVMxuku pUrwi kAlexani jagan AgrahaM vyakwaM ceSAru. Jagan was outraged as to why the work was not completed by the time of the flood.	Angry
వైఎస్ వివేకానందరెడ్డి, సుబ్బారెడ్డి, సంజీవరెడ్డిల మృతికి అసెంబ్లీ సంతాపం వ్యక్తంచేసింది. vivekAnaMxareVddi, subbAreVddi, saMjIvareVddila mqwiki aseVMblI saMwApaM vyakwaMcesiMxi. Assembly mourns the death of Vivekananda Reddy, Subba Reddy and Sanjeeva Reddy.	Sad
సన్‌రైజర్స్ భారీ స్కోరు. sanrEjars BArI skoru. Sunrisers scored a huge score.	Happy
నిజానికి హారర్ థ్రిల్లర్ సినిమాలంటే నాకు భయం. nijAniki hArar Wrillar sinimAlaMte nAku BayaM. In fact, I am afraid of horror thriller movies.	Fear
అయితే అనంతరం రెండు మ్యాచ్‌లు బంగ్లా, ఆఫ్ఘనిస్థాన్‌వంటి చిన్న జట్లపైఆడినవే. ayiwe anaMwaraM reVMdu myAclu baMglA, APGanisWAn vaMti cinna jatlapE Adinave. However, the next two matches were played against smaller teams like Bangladesh and Afghanistan.	No Emotion

Figure 5.2: Sample of annotated sentences for EI task in Telugu, WX and English translation

Sentence	Label
నేను ఈ షూటింగ్‌కు రాకముందు చాలామంది జేడీ గురించి బ్యాడ్‌గా చెప్పారు. nenu I RUtiMgku rAkamuMxu cAlAmaMxi jedI guriMci byAdgA ceVppAru. Before I got into this shooting, many people had told badly about the Jedi.	Yes
సంస్కృతి సంప్రదాయాలను మరచిపోకూడదని వెంకయ్య నాయుడు తెలిపారు. saMskqwi saMpraxAyAlanu maracipokUdaxani veVMkayya nAyudu weVlipAru. Venkaiah Naidu said that the culture and traditions should not be forgotten.	No

Figure 5.3: Sample of annotated sentences for HS task in Telugu, WX and English translation

5.3.4 Telugu Sarcasm Detection (SAR):

Motivated by the recent successful contextual word features, we created a sarcasm dataset for Telugu and empirically tested it with different embeddings.To the best of our knowledge, we are the first that leverage the annotations and consider the task of SAR for the Telugu language. For this task, annotators were required to select a label for sarcasm. The labels for sarcasm are “yes” and “no”. The examples in Telugu, WX, and English format for sarcasm are given in Figure 5.4.

Sentence	Label
తెలంగాణ రాష్ట్రం ఆర్థిక రంగంలో బాహుబలిలా దూసుకుపోతోంది. weVlaMgANa rARtraM ArWika raMgaMlo bAhubaliIA xUsukupowoMxi. Telangana State is doing well like bahubali in the financial sector.	Yes
ప్రతి నెల ఒకటో తేదీన జీతాలు వేర్లించాలని అన్నారు. prawi neVla oVkato wexIna jIwAlu ceVlliMcAlani annAru. He said salaries should be paid on the first day of every month.	No

Figure 5.4: Sample of annotated sentences for SAR task in Telugu, WX and English translation

5.3.5 Telugu Clickbait Detection:

Clickbait detection aims to identify the popular headlines that catch users' attention to read and click on the web link. Also, clickbait headlines typically provide sufficient information to make readers curious but insufficient to fulfill their curiosity, making them click the linked content. One of the reasons to attract user's attention is to generate revenue from the subscriptions, clicks, and views made by their readers [36]. We created a clickbait dataset for Telugu and empirically tested it with different embeddings. For this task, annotators were required to select a label for clickbait. The labels for clickbait are "yes" and "no". The examples in Telugu, WX, and English format for clickbait are given in Table 5.5.

Sentence	Label
వింత వాయిద్యం.. వినూత్న సంగీతం. viMwa vAyixyaM.. vinUwna saMgIwaM. Strange instrument .. innovative music.	Yes
నెఏన్స్‌లో టీన్!. nEntIslo tIn. Teen in the Nineties.	Yes
మనిషి చర్మంతో పుస్తకాలు. maniRi carmaMwo puswakAlu. Books with human skin.	Yes
ఆకాశం అంచున తాకి... AkASaM aMcuna wAki... The skies touched the edge ...	No

Figure 5.5: Sample of annotated sentences for clickbait task in Telugu, WX and English translation

To make the annotation process unbiased and with the motivation to produce the quality annotated datasets for Telugu, we handled the work with a third party. In this thesis, the annotation work for all the datasets was done by Elance IT Solutions Private Limited[12].

5.4 Annotation Tool

To annotate the data, we create a web-based user interface to simplify the task, as shown in Figure 5.6. Based on the guidelines, each user should read the sentence carefully and determine whether or not it expresses the sentiment, emotion, hate speech, sarcasm, or clickbait. If so, then categorize each task into the respective labels given in the guidelines. For example, in the SA task, the annotators need to select either "positive", "negative", or "neutral" based on the context and meaning of the words. All the responses were saved in the database. We consider majority voting out of five label responses for giving the final label for a sentence. To the best

[12] http://elancerits.com/

of our knowledge, we are the first that leverages the annotations and consider different NLP tasks for the Telugu language.

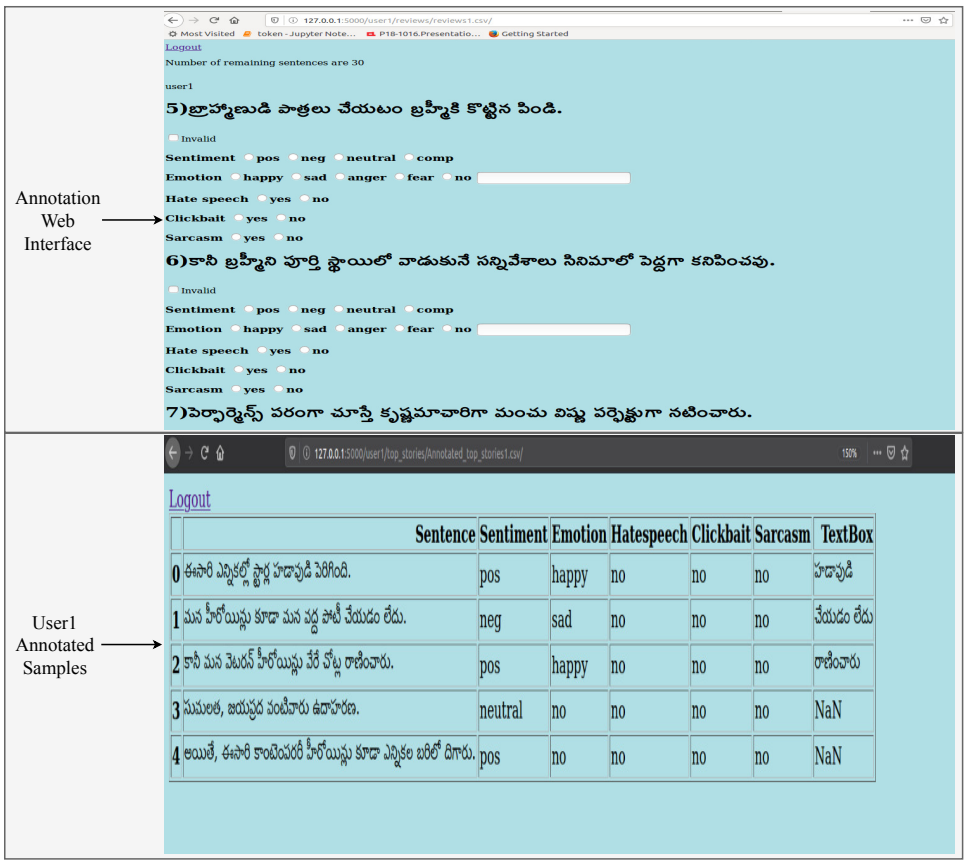


Figure 5.6: Figure shows a sample web interface for annotation process and example of user1 annotated sentences

Inter-Annotator Agreement: We made the five annotators work on a smaller dataset for verification as a first step to performing the annotation. The Fleiss' kappa score[13] was 0.95. The inter-annotator agreement on the whole dataset annotation was reported as 0.91.

5.5 Annotators Fair Compensation:

For all our annotation work, we provided the data to an *Elancer IT Solutions Private Limited*[14] company. To perform the annotation process for every task, *Elancer IT Solutions Private Limited* chose five native speakers of Telugu with excellent fluency, the company itself properly remunerates all the annotators.

[13]https://en.wikipedia.org/wiki/Fleiss%27_kappa
[14]http://elancerits.com/

5.6 Summary

In this chapter, we presented the general process of corpus creation, annotators guidelines, examples of Telugu annotated sentences for five NLP tasks and fair compensation to the annotators.

In the next chapter, we present our Multi-task Text GCN architecture that we developed using TEL-NLP dataset for SA, EI, HS and SAR. The work on clickbait dataset creation and detection is presented in chapter 8.

Chapter 6

TEL-NLP Dataset and Multi-task GCN for Text Classification

This chapter explains creating a novel architecture using GCN in a multi-task setting for Telugu. We developed an annotated TEL-NLP dataset for four different tasks viz. SA, EI, HS and SAR. The experiments show that our proposed MT-Text GCN improved the classification performance across the tasks. As we need a different set of syntactic features the fifth task clickbait detection is presented in chapter 6.

6.1 Background

To evaluate the performance of GCN with one of the Indian languages, Telugu, we analyze the GCN-based models with extensive experiments on four downstream tasks. In addition, we created an annotated Telugu dataset, TEL-NLP, for the four NLP tasks. Further, we propose a supervised graph reconstruction method, Multi-task Text GCN (MT-Text GCN) on the Telugu, that leverages to simultaneously (i) learn the low-dimensional word and sentence graph embeddings from word-sentence graph reconstruction using graph autoencoder (GAE) and (ii) perform multi-task text classification using these latent sentence graph embeddings. We argue that our proposed MT-Text GCN achieves significant improvements on TEL-NLP over existing Telugu pre-trained word embeddings [78], and multilingual pre-trained Transformer models: mBERT [34], and XLM-R [28]. On TEL-NLP, we achieve a high F1-score for four NLP tasks: SA (0.84), EI (0.55), HS (0.83) and SAR (0.66). Finally, we show our model's quantitative and qualitative analysis of the four NLP tasks in Telugu. We open-source our TEL-NLP dataset, pre-trained models, and code[1].

In summary, the main contributions of this chapter are as follows:

- We propose the multi-task learning model (MT-Text GCN) to reconstruct word-sentence graphs while achieving multi-task text classification with learned graph embeddings.

[1]https://github.com/scsmuhio/MTGCN_Resources

- TEL-NLP is the largest NLP dataset in Telugu, covering four NLP tasks (16,234 samples for SA, HS, SAR, and 9,675 samples for EI), unlike previous works, which focused only on a single task (SA) with a limited dataset of 2500 samples [86].

- For the Telugu language, we are the first to generate word embeddings using random walk based models: DeepWalk and Node2Vec, and Graph AutoEncoders (GAE).

- Our MT-Text GCN enabled better performance on Telugu when compared with multilingual pre-trained transformer models on four NLP tasks.

6.2 TEL-NLP Dataset Details

This section presents our dataset creation, TEL-NLP, for four NLP tasks such as SA, EI, HS, and SAR in Telugu. Here, we describe the dataset creation and annotation, inter-annotator agreement, and report the dataset statistics.

Dataset Creation and Annotation We sampled the 20k sentences from the existing large Telugu raw corpus [78] and given to Elance IT Solutions Private Limited[2] to perform the annotation task. Also, the annotation was carried out for four NLP tasks, viz. sentiment, emotion, hate-speech, and sarcasm. We create a web-based user interface to simplify the annotation task. Each user has to log in and do the annotation. The user interface shows the sentence and the respective labels for the four tasks. Five native speakers of Telugu with excellent fluency performed this annotation task. We provided the annotators with detailed definitions for each task with example sentences. All the annotators should read the sentence carefully and select the fitting label for that particular sentence. When the users click the submit button, their responses are saved in the database. To qualify and complete the annotation process, each annotator must correctly annotate at least 80% sentences from the sample dataset. After two rounds of sanity checks, we obtained 16,234 labeled sentences for SA, HS, SAR, and 9,675 for EI. We considered "positive" and "negative" labels for the SA task. "fear", "angry", "sad", and "happy" labels for the EI task, "yes" and "no" for HS and SAR datasets. As we considered the four basic emotions for the EI task, we got fewer samples when compared with other tasks.

Inter-Annotator Agreement We made the five annotators work on a smaller dataset for verification as a first step to performing the annotation. The Fleiss' kappa score[3] was 0.95. The inter-annotator agreement on the whole dataset annotation was reported as 0.91

TEL-NLP Description We performed our experiments on SA, EI, HS, and SAR datasets. The TEL-NLP dataset description and number of nodes in the graph are provided in Table 6.1.

[2] `http://elancerits.com/`

[3] https://en.wikipedia.org/wiki/Fleiss%27_kappa

Table 6.1: Statistics of the TEL-NLP dataset

Task	# Words	# Sentences	# Nodes	# Classes
SA	41501	16234	57735	2 (Positive: 9,488, Negative: 6,746)
EI	28447	9675	57735	4 (Fear: 121, Angry: 388) (Sad: 5,135, Happy: 4,031)
HS	41501	16234	57735	2 (Yes: 370, No: 15,864)
SAR	41501	16234	57735	2 (Yes: 402, No: 15, 832)

6.3 Proposed Method

In this section, we will explain (i) the overview of Graph Convolutional Network (GCN) [65] and Graph AutoEncoder (GAE), (ii) the proposed MT-Text GCN approach, and (iii) the procedure to build word-word, sentence-sentence, and word-word + word-sentence graphs.

6.3.1 GCN and Graph AutoEncoder

GCN is a multi-layer neural network that convolves a neighboring node's features and propagates its embedding vectors to its nearest neighborhoods. Consider a weighted, undirected graph denoted by $\mathcal{G} := (\mathcal{V}, \mathrm{A}, \mathrm{X})$, where $\mathcal{V}$ is a set of N nodes, and $\mathrm{A} \in \mathbb{R}^{NXN}$ is the symmetric adjacency matrix and $\mathrm{X} \in \mathbb{R}^{NXM}$ is the node feature matrix with each node dimension M. The Graph AutoEncoder [111] (GAE) obtains a reduced dimensional representation ($\mathrm{Z} \in \mathbb{R}^{NXK}$) of the input graph (A). Intuitively, starting from the node embedding, the GAE model can reconstruct an adjacency matrix $A^{'}$ close to the empirical graph (A), then the Z vectors should capture some essential characteristics of the graph structure. To reconstruct the graph, we stack an inner product decoder to this GAE as follows:

$$A^{'} = f(AH^{(1)}W_1) \tag{6.1}$$

$$Z = H^{(1)} = f(AXW_0) \tag{6.2}$$

where f denotes the activation function, $\mathrm{H}^{(1)}$ denotes the layer-1 output (latent representation), W_0 and W_1 denotes the weight parameters learned from the encoder and decoder model, respectively.

6.3.2 Multi-task Text GCN (MT-Text GCN)

Given an input graph A, our main objective is to build and train a model to reconstruct a graph (GAE) to learn word and sentence embeddings. These learned embeddings are used to perform multi-task text classification. Here, we perform multi-task text classification by providing sentence representations in two ways : (i) learned sentence embeddings from GAE, and (ii) each sentence is represented by taking the average of learned word embeddings from GAE. The proposed MT-Text GCN pipeline is shown in Fig 6.1. We use a 2-layer GCN of

which layer-1 acts as encoder and layer-2 as the decoder. The encoder (Z) and decoder output ($A^{'}$) using the layer-1 latent representations can be estimated from the Equations 6.1 and 6.4.

We use a LightGBM classifier to perform the multi-task text classification. Overall, the final objective function is given as

$$\mathcal{L} = \mathcal{L}_{MSE}(A, A^{'}) + \lambda \times \mathcal{L}_{MT_{CLA}}(\widehat{l}, l) \tag{6.3}$$

where, $\mathcal{L}_{MSE}$ (reconstruction loss) denotes the mean squared error (MSE) between the reconstructed graph $A^{'}$ and the empirical graph A. The sigmoid cross-entropy loss denoted by $\mathcal{L}_{MT_{CLA}}$ evaluate classification performance between predicted labels $\widehat{l}$ and ground truth labels (l). The λ parameter tunes the trade-off between reconstruction and classification performance. The model performance for different values of λ are reported in Fig 6.2.

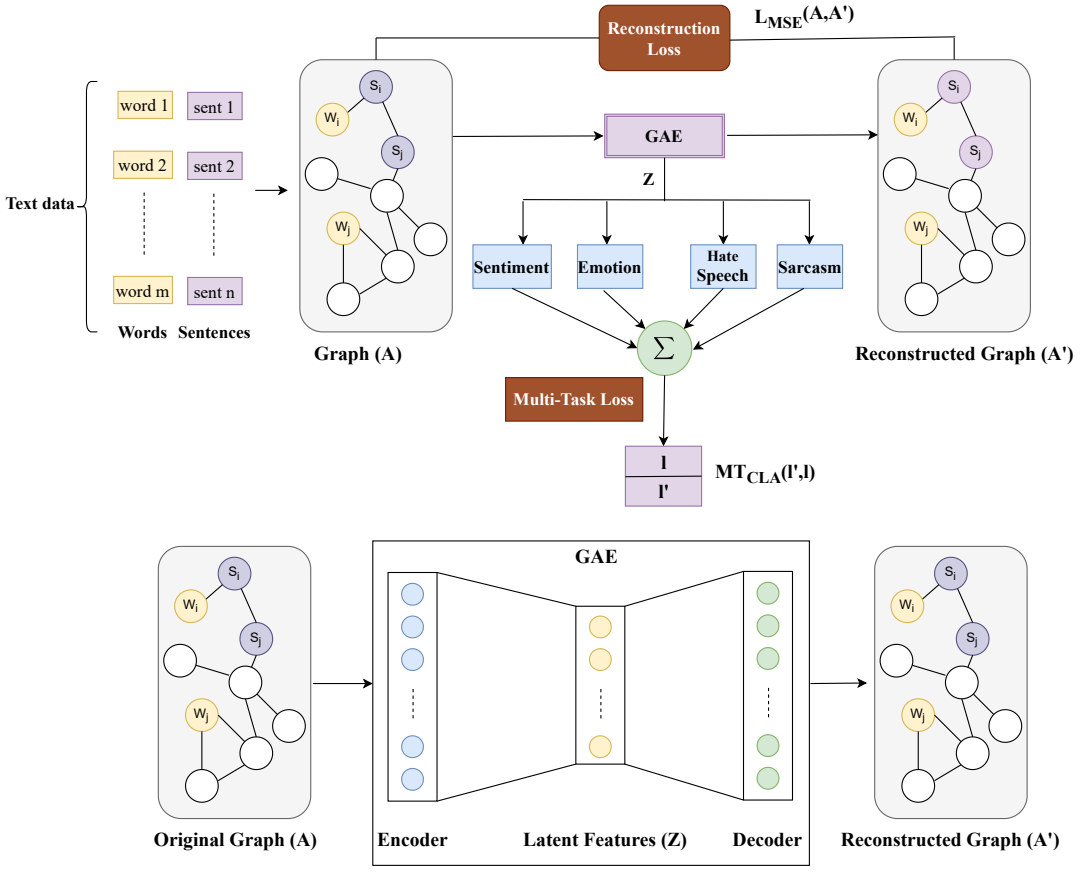


Figure 6.1: MT-Text GCN Pipeline: Rows in Z are low-dimensional node embeddings learned by the encoder from input graph A. LightGBM classifier outputs predicted labels $\widehat{l}$, and compares with actual labels l. $\mathcal{L}_{MSE}$ (reconstruction loss) denotes the mean squared error (MSE) between the reconstructed graph $A^{'}$ and the empirical graph A. The sigmoid cross-entropy loss denoted by $\mathcal{L}_{MT_{CLA}}$

6.3.3 Building Text Graph

We set the node feature vector (X) as an identity matrix, i.e., every word/sentence/word+sentence represents a one-hot vector, given as input to MT-Text GCN.

Word-level Graph (W): To construct a word-level graph, we explicitly calculated the global word co-occurrence matrix with a window size of 3, and the number of nodes is equal to the number of unique words in the vocabulary. Since point-wise mutual information (PMI) [30] yields better results than word co-occurrence count [134], we adopt PMI as a metric to calculate the weights between any two words. We use the global word-word co-occurrence matrix as the adjacency matrix (A), where the edge between the two words is calculated using PMI.

$$PMI(i,j) = log\frac{p(i,j)}{p(i)p(j)} \tag{6.4}$$

$$p(i,j) = \frac{\#W(i,j)}{\#W} \tag{6.5}$$

$$p(i) = \frac{\#W(i)}{\#W} \tag{6.6}$$

where #W(i, j) is the number of sliding windows that contain words i and j, #W(i) is the number of sliding windows that contain the word i, #W is the total number of sliding windows in the dataset.

Sentence-level Graph (S): In this graph, nodes represent the sentences, and we use pre-trained word embeddings (Word2Vec) to obtain sentence vector representation by averaging all the word vectors of the sentence. We use cosine distance to calculate the similarity score between two nodes.

Word+Sentence-level Graph (W+S): In this graph, words and sentences act as nodes, we adopt the global word co-occurrence statistics and TF-IDF frequencies to fill the edge weights between the nodes.

$$A_{ij} = \begin{cases} \text{PMI}(i,j), & \text{i, j are words} \\ \text{TF-IDF}_{ij}, & \text{i is word, j is sentence} \\ 1, & i = j \\ 0, & \text{otherwise} \end{cases} \tag{6.7}$$

6.4 Experimental Setup and Results

This section describes different features such as traditional features, pre-trained word embeddings, random walk-based representations, and MT-Text GCN model training setup. This chapter uses Light Gradient Boosting Method (LightGBM) as a classifier to train the models for all the feature representations.

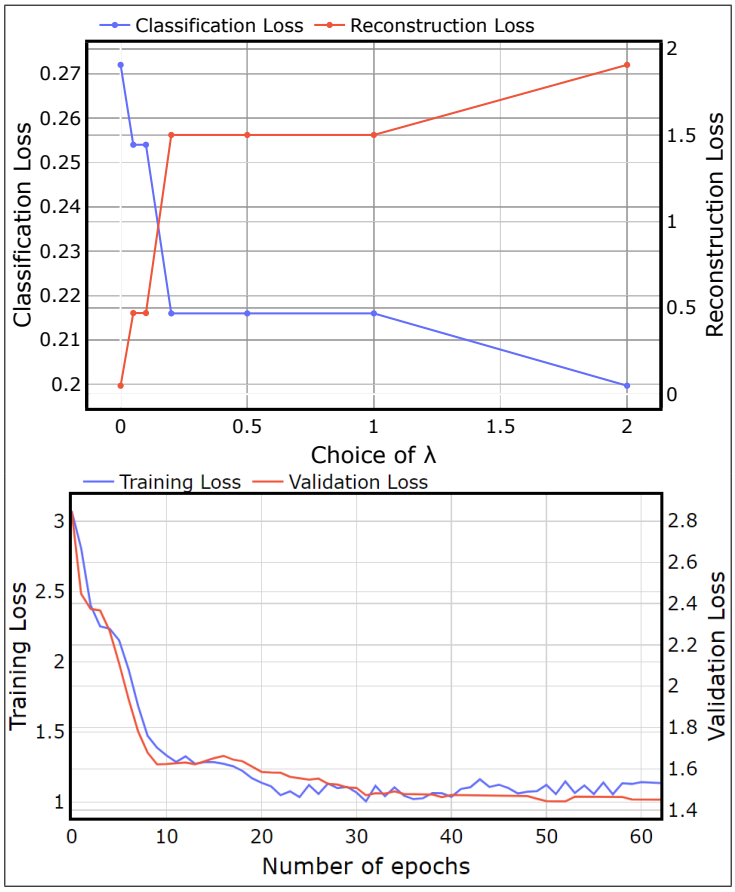


Figure 6.2: Example of reconstruction (red)/ classification (blue) loss with different λ (top), train and validation learning curves shows a good fit at $\lambda = 0.2$ (bottom). Each point on the curve is the average of loss with corresponding λ

6.4.1 Baselines

We compare our MT-Text GCN with traditional features, distributed word representations, and random walk based methods.

Traditional Features We use the traditional Bag-of-words (BoW) [126] and TF-IDF features [105] as our baseline methods to perform text classification on four NLP tasks. The BoW model uses the frequency of words in a sentence as a feature. The BoW model with term frequency times inverse document frequency of words in a document is used as the TF-IDF feature.

6.4.2 Telugu Word Embeddings

We use the existing pre-trained Telugu word embeddings [78] to perform the text classification on our dataset, TEL-NLP: (1) A **Te-Word2Vec (Te-W2V)** model trained on a large Telugu corpus with 300-hidden dimensions as word representations. (2) **Te-GloVe** based word

vectors (each word is a 200-dimension vector). (3) A 200-dimension **Te-FastText (Te-FT)** embeddings. (4) **Te-Meta-Embeddings (Te-ME)**, an ensemble of Telugu Te-W2V, Te-GloVe, and Te-FT embeddings are used to create Te-ME of 300 dimension [78].

6.4.3 Deep Learning Methods

CNN: Proposed by [62], we perform convolution and max-pooling operations on Telugu word embeddings to get a representation of text that is used for text classification.
LSTM: Defined in [119, 74], we use the last hidden state as text representation to perform text classification. We train LSTM with five input representations (i) One-Hot encoding and (ii) the four pre-trained Telugu word embeddings [78].

6.4.4 Random Walk based Representations

DeepWalk (DW): Proposed by [96], this approach uses local information obtained from truncated continuous feature representations random walks, equivalent to sentences to learn latent representations of nodes in a graph.
Node2Vec (N2V): A framework for learning nodes in the graph by using a biased random walk procedure, which efficiently explores diverse neighborhoods in [45]. To accommodate the morphological richness in Indian languages like Telugu, we consider the different small window sizes (2,3,4) for creating the co-occurrence matrices.

6.4.5 Graph based Representations

GCN-W: A word-level graph CNN model that operates convolution over word-word adjacency matrix [134], in which first-order approximation of Chebyshev filter is used.
GCN-S: A sentence-level graph CNN model that operates convolution over sentence embeddings similarity matrix [147], in which first-order approximation of Chebyshev filter is used.
GCN-W+S: A word+sentence level graph CNN model that operates convolution over word-word and word-sentence embeddings similarity matrix [134], in which first-order approximation of Chebyshev filter is used.

6.4.6 Model Setup

To perform text classification using GCN, we set the first convolution layer's embedding size as 200 and the input feature vector as a one-hot vector for different GCN models such as GCN-W, GCN-S, GCN-W+S, and MT-Text GCN. We use the Adam optimizer [63] with an initial learning rate of 0.001, RELU as the activation function at the first GCN layer, and L2 weight decay equal to $5e^{-4}$. We applied the dropout with a probability of 0.5 after the first GCN layer and trained all the GCN models for a maximum of 100 epochs. We set the callbacks

and checkpoints in which the model training stops automatically if the validation loss does not decrease for ten consecutive epochs. For the GCN-W model, we set the context window size as three while constructing a word-word co-occurrence matrix using PMI as an input adjacency matrix. Here, we jointly perform the reconstruction of a graph using GAE and the latent node representations from GAE to build the classification model. The experiments were performed on a single V100 16GB RAM GPU machine.

To determine the choice of λ, we perform an extensive grid search and the corresponding results are shown in Fig 6.2. As the λ increases, the reconstruction performance deteriorates, and classification loss improves. When $\lambda = 0$, the reconstruction loss yields the best score, whereas classification loss is peak. We chose $\lambda = 0.2$ based on the performance of both reconstruction and classification loss. Fig 6.2 depicts the model training vs. validation loss performance when passing the parameter λ =0.2. Here the first layer embedding dimensions are 200. Fig 6.2 tells us that both training and validation loss decreased as the number of epochs increased. We also experimented with GCN-W and GCN-S on a single task for different embedding dimensions, and corresponding F1 scores for each model are shown in Table 6.2.

Dataset Splitting: We followed the 5-fold cross-validation split for all the methods. One fold was used for the test (20% data), and the 4-fold data was split into train/validation (70/10). We calculated the average of 5-folds and reported the results in Table 6.2.

Classification Metrics: Table 6.2 reports each task classification results, such as F1-score for various models, except for the HS task. Since our HS task data has a class imbalance issue, we report the weighted F1 score.

6.5 Results and Analysis

The effectiveness of our proposed model is evaluated by comparing it with traditional features, different word embeddings [78], deep learning methods, node embeddings, and existing pre-trained multilingual transformer models mBERT [34] and XLM-R [28]. The results can be analyzed from the Table 6.2 as follows:

Traditional Features: We considered the traditional feature representations BoW and TF-IDF generated from the Telugu corpus as the baselines. Since our baseline features do not capture the semantic and contextual information, the baseline system achieves lower F1 scores for all tasks, as shown in Table 6.2, block (A).

Distributed Word-Embeddings: For each text classification task, we used the pre-trained word-embeddings generated from the Telugu corpus (Te-W2V, Te-GloVe, Te-FT, and Te-ME) [78] as input, and the results are reported in Table 6.2. With the local (Te-W2V, Te-FT) and global (Te-GloVe) contextual word embeddings, the system achieves an improved F1 score compared to the baseline model for all the four NLP tasks in which Te-FT embeddings have shown better performance than other word-embeddings, as shown in Table 6.2, block (B). How-

ever, from Table 6.2, we observe that the performance of sarcasm task results is deficient and equal to random guessing due to the high class imbalance problem.

Table 6.2: TEL-NLP results on four tasks in terms of F1-score using: (A) traditional features, (B) distributed word embeddings, (C) deep learning representations, (D) random walk embeddings, (E) pre-trained multi-lingual transformers, and (F) our proposed method: MT-Text GCN and its variations. All these methods were trained on Telugu corpus. Since HS data has a class imbalance issue, we report the weighted F1-score

	Tasks→	SA	EI	HS	SAR
Block	Feature set↓	F1-score	F1-score	F1-score	F1-score
(A)	BoW	0.60	0.32	0.56	0.50
	TF-IDF	0.61	0.32	0.58	0.50
(B)	Te-W2V	0.79	0.44	0.63	0.50
	Te-GloVe	0.79	0.43	0.68	0.52
	Te-FT	0.83	0.50	0.66	0.55
	Te-ME	0.82	0.48	0.68	0.50
(C)	CNN	0.64	0.30	0.63	0.51
	LSTM	0.81	0.45	0.71	0.53
	LSTM-TeW2V	0.75	0.49	0.68	0.52
	LSTM-TeGloVe	0.66	0.35	0.67	0.51
	LSTM-TeFT	0.72	0.41	0.67	0.53
	LSTM-TeME	0.65	0.38	0.68	0.52
(D)	DW	0.56	0.30	0.70	0.50
	N2V-WS2	0.77	0.48	0.66	0.54
	N2V-WS3	0.78	0.47	0.80	0.55
	N2V-WS4	0.78	0.46	0.82	0.58
(E)	mBERT	0.68	0.42	0.50	0.50
	XLM-R	0.64	0.34	0.50	0.50
(F)	ST-TextGCN-W	0.69	0.36	0.75	0.64
	ST-Text GCN-S	0.77	0.43	0.82	0.65
	ST-Text GCN-W+S	0.82	0.48	0.83	**0.67**
	MT-Text GCN-W+S	**0.84**	**0.55**	**0.83**	0.63

Deep Learning Methods: Although the deep learning models such as CNN and LSTM generated from Telugu corpus outperform the baseline setting, CNN with randomly initialized word embedding yields a lower F1-score when compared with different word embeddings, displayed in Table 6.2, block(C). The LSTM-based model outperforms the traditional features and performs better than CNN. Moreover, the sequence of tokens with one-hot representation as input to LSTM yields a better F1 score than four pre-trained word embeddings, except for the EI task. These results infer that LSTMs do not address the long-term dependency problems in morphologically rich languages, Telugu, which affects the classification performance of four tasks.

Random Walk based Approaches: From Table 6.2, block(D) describes the results of random walk based models (DW / N2V) generated from the Telugu corpus as input to the LightGBM model for four tasks. We make the following observations from Table 6.2: (i) the DW embeddings report a low F1-score for four tasks when compared to all the methods except the baseline setting in Table 6.2. (ii) We observe that the N2V embeddings learned from the three context graphs (i.e., varying the different context window sizes: 2, 3, and 4) given as an input, the model displays an increasing F1-score and reports better results with a context window size of 3 & 4. (iii) Compared to distributed word embeddings and deep learning methods, N2V-WS3 and N2V-WS4 display higher F1 scores for the HS and SAR tasks. (iv) These results demonstrate that N2V is ideal for learning both local and global neighbors so that it can construct

MT-Text GCN-W+S	Te-W2V	Te-GloVe	Te-FT
(తప్పుకోవడం, Abandonment)	(బరిలో, Ring)	(పోటీల్లో, Competition)	(పోటీల్లో, Competition)
(పాల్గొన్నాను, Participated)	(రేసులో, Race)	(సాధించారు, Steal)	(బరిలో, Ring)
(గెలవగానే, Won)	(వెనుకంజలో, Trailing)	(బరిలో, Ring)	(పోటీలోని, Competition)
(రేసులో, Race)	(ధీమాతో, Confident)	(అర్హత, Eligible)	(పోటీలో, Competition)
(విజయం, Success)	(లీడింగ్లో, Leading)	(విజేతగా, Winner)	(పోటీల్లో, Competitions)
(విరుచుకుపడి, Destruction)	(ఆధిక్యతలో, Leading)	(గెలిచిన, Won)	(అభ్యర్థులుగా, Candidates)
(కుర్రకారుని, Boys)	(కంచుకోటగా, Stronghold)	(పాల్గొనే, Participant)	(రేసులో, Race)
(ప్రయత్నిస్తాడు, Tries)	(ధీమాలో, Tries)	(స్థానం, Position)	(గెలిచినవారిలో, Winner)

Figure 6.3: Top 8 similar words for the word "competition" using (i) MT-Text GCN-W+S (ii) Te-W2V (iii) Te-GloVe and (iv) Te-FT

more semantic vertex neighborhoods, and it is better at classifying the text classification tasks on the imbalanced datasets (HS and SAR), while DW has the limitation of no control over explored neighborhoods.

pre-trained Transformers: We evaluate whether fine-tuning the existing pre-trained multilingual transformer models mBERT [34], and XLM-R [28] is useful for adapting each of the NLP tasks. From Table 6.2, we find that fine-tuning results show better performance than baseline, CNNs, and DW methods. However, the performance is lower compared to Telugu pre-trained word embeddings, N2V embeddings, and our proposed MT-Text GCN reported in Table 6.2, block(E). The average length of sentences in these Telugu NLP tasks is much longer than that in English datasets. Moreover, the graph is constructed using word-sentence statistics, meaning long texts may produce more sentence connections transited via an intermediate word node. This potentially benefits graph mechanisms, leading to better performances with MT-Text GCN multilingual transformers.

6.5.1 Graph based Node Embeddings

ST-Text GCN Results: We present the single task text classification results for models trained on word-word (ST-Text GCN-W), sentence-sentence (ST-Text GCN-S), and word-word + word-sentence (ST-Text GCN-W+S) graphs in Table 6.2, block(F). (i) From Table 6.2, we infer that sentence embeddings learned from ST-Text GCN-S show a better classification performance across four individual tasks compared to word embeddings obtained from ST-Text

GCN-W. (ii) Further, the single task text classification model trained on the W+S graph achieved high performance compared to ST-Text GCN-W and ST-Text GCN-S. (iii) These results indicate that graph-based models learned better contextual representation for each sentence rather than sentence representation obtained from an average of word embeddings. (iv) Unlike ST-Text GCN-S, we are not providing any edge information between a sentence to sentence. However, we observe that the latent sentence embeddings learned from ST-Text GCN-W+S reports better than ST-Text GCN-S in Table 6.2. (v) Although EI and SAR tasks have a data imbalance problem, all the GCN models yield better results than other feature representations. We reported that the quantitative analysis of each task is displayed in Fig 6.4.

MT-Text GCN Results: Like ST-Text GCN-W+S, we use the sentence embeddings as input to the classification model when training the MT-Text GCN-W+S. Table 6.2 shows that (i) the classification performance of SA and HS tasks has increased the performance in the multi-task setting. (ii) The classification performance of the SAR task was deficient, possibly due to class imbalance and lack of solid correlations across some task pairs. (iii) Overall, from Table 6.2, we observe that GCN-based models outperform all the tasks except a little less for the SAR task.

6.5.2 Qualitative Analysis

This study tried to understand how MT-Text GCN helps in regularizing word embeddings. Since word vectors are numerical representations of contextual similarities between the words, they can be manipulated and made to perform unusual tasks like finding the degree of similarity between two words [80]. To verify the quality of generated Telugu word-embeddings, we extract the top-8 semantically related words for "(competition)" using Te-W2V, Te-GloVe, Te-FT, and compared with MT-Text GCN-W+S, as shown in Fig 6.3. From Fig 6.3, we can observe that the semantically related words obtained for MT-Text GCN-W+S are similar to Te-W2V, Te-GloVe, and Te-FT.

6.5.3 Quantitative Analysis

Fig 6.4 showcase the classification performance (F1-score) of the models ST-Text GCN-W and ST-Text GCN-S on SA, EI, HS, and SAR datasets with different embedding dimensions at the first GCN layer. From Fig 6.4, we can observe that ST-Text GCN-S shows better classification performance and an increasing F1-score with different embedding dimensions when compared to ST-Text GCN-W. These results infer that the increase in the dimension yields an improved accuracy over all the tasks.

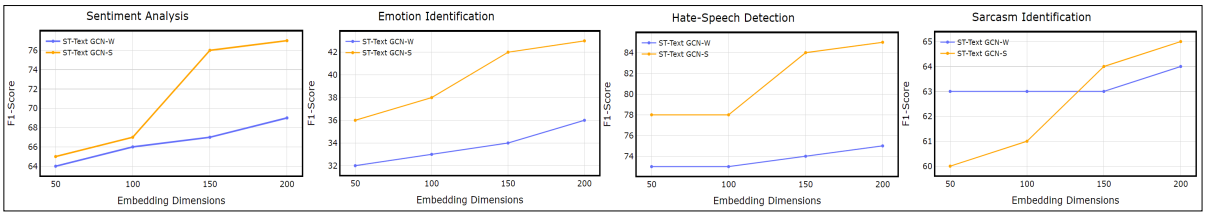


Figure 6.4: Test accuracy results for four tasks (i) SA (ii) EI (iii) HS and (iv) SAR by varying embedding dimensions

(a)

Actual \ Predicted	Negative	Positive
Negative	1074	274
Positive	229	1668

(b)

Actual \ Predicted	Angry	Fear	Happy	Sad
Angry	6	0	9	52
Fear	0	2	5	11
Happy	3	2	644	140
Sad	15	2	109	841

(c)

Actual \ Predicted	No	Yes
No	2613	569
Yes	8	57

(d)

Actual \ Predicted	No	Yes
No	3103	13
Yes	57	74

Figure 6.5: Confusion matrices of MT-Text GCN-W+S for three best performing tasks: (a) SA (b) EI (c) HS and (d) SAR using ST-Text GCN-W+S

6.5.4 Error Analysis

We analyzed the error cases in detail for the class labels that belong to each task. We observed from Fig 6.5a that for the SA task: (i) 20% of the negative class samples are predicted as positive, and (ii) 12% of the positive class samples are predicted as negative. For EI, we observe from the Fig 6.5b that (i) 78% of angry class samples, 65% of fear class samples, 17% of happy class samples are predicted sad, and (ii) 11% of sad class samples are predicted happy. These results infer that our system is more biased towards the Sad class samples due to the class imbalance problem in Angry and Fear class samples. Fig 6.5c shows the confusion matrix for the HS task, and we make the following observations: (i) The recall of yes (88%) and no (82%) classes are much higher, (ii) our model makes a higher percentage of false negatives (i.e., 18% of no class samples are predicted yes). This is mainly due to the class imbalance issue, and we report the weighted F1 score for HS task data in the results Table 6.2. For the SAR task,

we observe that (i) 43% of yes class samples are predicted no (i.e., our system makes more false positives), and (ii) the presence of false negatives (0.4%) is low compared to the HS task, as shown in Fig 6.5d.

Task	Sentence	Act	Pred	Reason
SA	ఇదే గనుక మరో సారి రిపీట్ అయితే నాగపూర్‌లో పరుగుల వరద ఖాయం. ixe ganuka maro sAri ripIt ayiwe nAgapUrlo parugula varaxa KAyaM. **If the same repeats again, there will be a flood of runs in Nagpur.**	Pos	Neg	**Unable to recognise idiomatic phrase.**
SA	రేషన్ బియ్యం పక్కదారి పట్టకుండా పండాతం కార్డులను ఆధార్‌తో అనుసంధానం చేశామన్నారు. reRan biyyaM pakkaxAri pattakuMdA vaMxaSAwaM kArdulanu AXArwo anusaMXAnaM ceSAmannAru **All the ration cards are linked to Aadhar 100% to avoid any malpractices in the distribution of rice.**	Pos	Neg	**Unable to recognise noun phrase ellipsis.**
EI	భారత్ రెండో ఇన్నింగ్స్ సందర్భంగా బ్యాటింగ్ చేస్తున్న పంత్‌ను వ్యక్తిగతంగా దూషిస్తూ రెచ్చగొట్టాడు. BAraw reVMdo inniMgs saMxarBaMgA byAtiMg ceswunna paMwnu vyakwigawaMgA xURiswU reVccagoVttAdu. **He provoked Pant who was batting during India's second innings with personal abuses.**	Angry	Sad	**Unable to parse complex syntax**: Telugu sentence had more verbs (finite and non-finite) compared to English which makes it complex for our model to understand.
EI	అంతర్జాతీయ చమురు ధరలు రోజు రోజుకు పెరుగుతున్నాయి aMwarjAwIya camuru Xaralu roju rojuku peVruguwunnAyi. **International oil prices are rising day by day.**	Sad	Happy	**Misunderstood the implicature** associated with the word "increase" which has negative sense in this context.
HS	కానీ పూర్తి ఫిట్‌నెస్ సాధించకుంటే అతడిని ఐపీఎల్‌కు పంపము. kAnI pUrwi PitneVs sAXiMcakuMte awadini EpIeVlku paMpamu. **We won't send him to the IPL until he is not fully fit.**	No	Yes	**Model got confused** with the presence of **multiple negative words**.
HS	సోషల్ మీడియా వేదికగా చివాట్లు పెడుతున్నారు. soRal mIdiyA vexikagA civAtlu peVduwunnAru. **The social media platform is being reprimanded.**	Yes	No	**Misunderstood the implicature** associated with the word "reprimanded" which has positive sense in this context.
SAR	న్యూజిలాండ్ బౌలర్లు తొలి మ్యాచ్‌లో శ్రీలంకను 136 పరు గులకే కుప్ప కూల్చారు. nyUjilAMd bOlarlu woVli myAclo SrIlaMkanu 136 paru gulake kuppa kUlcAru. **New Zealand bowlers thrashed Sri Lanka by 136 runs in the first match.**	Yes	No	**Unable to understand the metonymy** associated with the team phrase.
SAR	బాల్‌టాంపరింగ్‌లో అడ్డంగా దొరికిపోయాడు. bAltAMpariMglo addaMgA xoVrikipoyAdu. **Got caught red-handed in ball tampering.**	No	Yes	**Unable to understand idiomatic phrase**.

Figure 6.6: MT-Text GCN-W+S: Wrong predictions by the model on four text classification tasks: SA, EI, HS, SAR

Fig 6.6 displays the failure cases for four tasks, and we make the following observations: (i) The model could not recognize “idiomatic” and “noun phrase ellipses” in the SA task. (ii) For the EI task, the model misunderstood the word “increase” in a positive sense in this context and was unable to parse sentences with “complex syntax”. For instance, sentence 3 in Fig 6.6 shows that the relative clause of English gets translated to an adjectival construction, and the noun phrase in the prepositional phrase gets verbalized in Telugu. Moreover, the Telugu sentence had more verbs (finite and non-finite) than English, making it complex for our model to understand. (iii) In the HS task, the model got confused with positive words and also misunderstood the word “reprimanded” in a negative sense in this context. (iv) Like the SA task, the model could not recognize “idiomatic” or “metonymy” in the SAR task.

6.6 Summary

In this chapter, we propose an MT-Text GCN to jointly (i) perform graph reconstruction, (ii) learn word embeddings, (iii) learn sentence embeddings, and (iv) do multi-task text classification on four significant NLP tasks - SA, EI, HS, and SAR. The experimental results show that our proposed model reports an improved classification performance across three tasks. In the future,

we aim to create pre-trained transformer models for Telugu. We believe that this work will pave the way for better research in resource-poor languages.

In the next chapter, we present our main work where we tried to solve the low resource problem of Telugu. We present an extended dataset to TEL-NLP along with pre-trained transformer feature representations.

Chapter 7

Datasets, Embeddings, and Models for Text Classification Tasks

This chapter tries to solve the low resource problem of Telugu by creating corpus, different annotated datasets for four NLP tasks viz. SA, EI, HS and SAR. We created several pre-trained feature representations with the help of Telugu raw corpus. We made all the resources publicly available to other researchers.

7.1 Background

Due to the lack of a large annotated corpus, many resource-poor Indian languages struggle to reap the benefits of recent deep feature representations in Natural Language Processing (NLP). Moreover, adopting existing language models trained on large English corpora for Indian languages is often limited by data availability, rich morphological variation, syntax, and semantic differences. In this chapter, we explore the traditional to recent efficient representations to overcome the challenges of low resource language, Telugu. In particular, our main objective is to mitigate the low-resource problem for Telugu: Overall, we present several contributions to a resource-poor language viz. Telugu. (i) a large annotated data (35,142 sentences in each task) for multiple NLP tasks such as sentiment analysis, emotion identification, hate-speech detection, and sarcasm detection, (ii) we create different lexicons for sentiment, emotion, and hate-speech for improving the efficiency of the models, (iii) pre-trained word and sentence embeddings, and (iv) different pre-trained language models for Telugu such as *ELMo-Te*, *BERT-Te*, *RoBERTa-Te*, *ALBERT-Te*, and *DistilBERT-Te* on a large Telugu corpus consisting of 80,15,588 sentences (16,37,408 sentences from Telugu Wikipedia and 63,78,180 sentences crawled from different Telugu websites). Further, we show that these representations significantly improve the performance of four NLP tasks and present the benchmark results for Telugu. We argue that our pre-trained embeddings are competitive or better than the existing multilingual pre-trained models: *mBERT*, *XLM-R*, and *IndicBERT*. Lastly, the fine-tuning of pre-trained models show higher performance than linear probing results on four NLP tasks with following F1-scores: Sentiment (68.72), Emotion (58.04), Hate-Speech (64.27) and Sarcasm

(77.93). We also experiment on publicly available Telugu datasets (Named Entity Recognition, Article Genre Classification, and Sentiment Analysis), find that our Telugu pre-trained language models (*BERT-Te* and *RoBERTa-Te*) outperform the state-of-the-art system except for the sentiment task. We open-source our corpus, four different datasets, lexicons, embeddings, and code `https://github.com/Cha14ran/DREAM-T`. The pre-trained transformer models for Telugu are available at `https://huggingface.co/ltrctelugu`.

The main contributions of this work are as follows:

- We created a large Telugu corpus of 80,15,588 sentences (16,37,408 sentences from Telugu Wikipedia and 63,78,180 sentences crawled from different Telugu websites).
- We create a large annotated data of 35,142 sentences for the four NLP tasks in the Telugu language, in contrast to earlier works experimented with tiny data for a single task.
- We build sentiment, emotion, and hate-speech lexicons at the word level and train the classifiers to estimate the performance as a baseline.
- We create pre-trained language models for Telugu such as *ELMo-Te*, *BERT-Te*, *RoBERTa-Te*, *ALBERT-Te*, and *DistilBERT-Te* on our large Telugu corpus (80,15,588 sentences).
- We create the benchmark results empirically for four NLP tasks in the Telugu language.
- We outperform the multilingual pre-trained language models such as *mBERT* [34], *XLM-R* [70], *Sub-word Byte Pair* [48], *Facebook FastText* [13], and *IndicBERT* [57].

7.2 Dataset Description

This section presents the process of collecting the Telugu corpus, pre-processing steps, and creating annotated data for four NLP tasks.

7.2.1 Corpus Collection Process

We manually identified a few online Telugu websites that publish movie reviews and news related to business, politics, sports, national and international. Data is collected from Telugu websites and the links to each website is mentioned in chapter 3, section 5.1. From these websites we crawled our corpus of 63,78,180 sentences. we added the available Telugu Wikipedia dump of 16,37,408 sentences and the total Telugu corpus consists of 80,15,588 sentences.

7.2.2 Data Pre-processing

The following steps were taken to ensure good quality input data for our models.

Text	Conversion
Money	“10.46” - “10:46”
Abbreviations	“E.D” - “E:D”
	“S.S.C” - “S:S:C”
Units of Measurement	“k.m” - “k:m”
	“9.2” - “9:2”
Overs & Run Rate	“4.5” - “4:5”
Domains	“.com” - “:com”

Table 7.1: Pre-processing of text using normalization

Typo-Error Sentence	ఈ సందర్భంగా నవ్యాంధ్ర ఫిలిం ఛాంబర్ అధ్యక్షుడు ఎస్.వి.ఎన్.రావు మాట్లాడుతూ. అచ్చమైన తెలుగు టైటిల్ తో రూపొందుతున్న ఈ సినిమా ట్రయిలర్ బాగుంది.
English Notation	Navyandhra Film Chamber President S.V.N. Rao spoke on this occasion. The trailer of this movie which is being made with a perfect Telugu title is good
Split Sentence	ఈ సందర్భంగా నవ్యాంధ్ర ఫిలిం ఛాంబర్ అధ్యక్షుడు ఎస్. వి. ఎన్. రావు మాట్లాడుతూ. అచ్చమైన తెలుగు టైటిల్ తో రూపొందుతున్న ఈ సినిమా ట్రయిలర్ బాగుంది.
English Notation	On this occasion Navyandhra Film Chamber President S. V. N. Rao speak. The trailer of this movie which is being made with a perfect Telugu title is good.
Corrected Sentence	ఈ సందర్భంగా నవ్యాంధ్ర ఫిలిం ఛాంబర్ అధ్యక్షుడు ఎస్. వి. ఎన్. రావు మాట్లాడుతూ, అచ్చమైన తెలుగు టైటిల్ తో రూపొందుతున్న ఈ సినిమా ట్రయిలర్ బాగుంది.
English Notation	On this occasion Navyandhra Film Chamber President S. V. N. Rao said that the trailer of this movie which is being made with a proper Telugu title is good.

Figure 7.1: Pre-processing of broken sentences by correcting the type errors

- Handling of broken sentences: Indic Tokenizer[1] is used for splitting the paragraph into sentences. We observed a few broken sentences due to typo errors, as shown in Figure 7.1. We manually merged these sentences.

- Text normalization: The text normalization includes expanding abbreviations, conversion of unit measurements, run rates, etc., changes reported in Table 7.1.

- Sentences with no information: We removed sentences that contain one or two words. Those include sentences with only names, locations, or dates, as they do not provide any information.

- Imbalanced data: We partially addressed this with data chosen from multiple domains with an appropriate number of samples. Further, we experimented with over and under-sampling techniques to handle class imbalance present in each task.

- Removal of special characters: We removed the special characters like arrow marks and Chinese characters from the corpus.

7.2.3 Data Statistics

The detailed statistics of the data in each domain for each task are tabulated in Table 7.2. From Table 7.2, we observe that in the movie domain, we have 9,882 sentences. In the political domain, we have 14,306 sentences and 10,954 sentences present in the sports domain.

[1] https://github.com/anoopkunchukuttan/indic_nlp_resources

Table 7.2: Dataset Statistics for the four NLP tasks

Tasks	# Total Instances	# Domains	# Classes	# Instances	Top Classes
Sentiment	35142	movies	3	9882	Neutral(5344),Pos(3135),Neg(1403)
		politics		14306	Neutral(7640), Pos(3461), Neg(3205)
		sports		10954	Neutral(5924),Pos(2892),Neg(2138)
Emotion		movies	5	9882	Neutral(7679), Happy(1336), Sad(776), Angry(47), Fear(44)
		politics		14306	Neutral(9869), Sad(2604), Happy(1539), Angry(245), Fear(49)
		sports		10954	Neutral(7919),Sad(1755),Happy(1156),Angry(96),Fear(28)
Hate-speech		movies	2	9882	No(9845), Yes(37)
		politics		14306	No(14076), Yes(230)
		sports		10954	No(10851), Yes(103)
Sarcasm		movies	2	9882	No(9778), Yes(104)
		politics		14306	No(14118), Yes(188)
		sports		10954	No(10844), Yes(110)

Table 7.3: Statistics of lexicons in each NLP task

Task	#Lexicons
Sentiment Analysis	positive (3846), negative(4501)
Emotion Identification	anger (351), fear (118), happy (3763), sad (4746)
Hate-speech	hate (155)

7.3 Feature Representation Methods

To simulate the models, we built the below hand-crafted and automated feature representations on Telugu corpus (80,15,588 sentences) from scratch. The following set of features was implemented to use in our four tasks.

7.3.1 Baseline Feature Representations

The three traditional feature representation methods like lexicon [50, 133, 43], Bag-Of-Words (BoW) [126], and Term Frequency-Inverse document Frequency (TF-IDF) [105] are successful in many of the NLP tasks. Although these methods are simple to implement, yet produce better results. We implement these on our Telugu corpus. We consider these three feature representation methods our baseline compared with created embeddings for Telugu from further subsections. Below, we discuss these traditional methods.

Table 7.4: Sample of positive sentiment lexicons in Telugu, WX, and English

Telugu_word	WX_word	English_word
మంచి	maMci	Good
నిజాయితీ	nijAyiwI	Honest
కృతజ్ఞతలు	kqwajFawalu	Thanks
న్యాయాన్ని	nyAyAnni	Justice
సంతోషం	saMwoRaM	Joy
మద్దతు	maxxawu	Support
విజయం	vijayaM	Success
ఆశ	ASa	Hope
విజయ	vijaya	Victory
గెలిచి	geVlici	Won

Table 7.5: Sample of negative sentiment lexicons in Telugu, WX, and English

Telugu_word	WX_word	English_word
చెడు	ceVdu	Bad
అన్యాయాన్ని	anyAyAnni	Injustice
అవిశ్వాస	aviSvAsa	Antitrust
అనుమానం	anumAnaM	suspected
అసభ్య	asaBya	Rude
అక్రమంగా	akramaMgA	Illegally
ఫోర్జరీకి	PorjarIki	Forgery
మృతి	mqwi	Death
ఓటమి	otami	Defeat
విఫలం	viPalaM	Fail

Table 7.6: Sample of happy emotion lexicons in Telugu, WX, and English

Telugu_word	WX_word	English_word
విజయం	vijayaM	Success
రాణించారు	rANiMcAru	Excelled
మంచి	maMci	Good
హర్షం	harRaM	Elation
అదృష్టంగా	axqRtaMgA	Good luck
ఛాంపియన్గా	CAMpiyangA	Champion
అగ్రస్థానంలో	agrasWAnaMlo	Top
సంచలనం	saMcalanaM	Buzz
ఆనందదాయకం	AnaMxaxAyakaM	Pleasure
అద్భుతం	axBuwaM	Awesome

Table 7.7: Sample of angry emotion lexicons in Telugu, WX, and English

Telugu_word	WX_word	English_word
సీరియస్	nippulu	Serious
మండిపడ్డారు	maMdipaddAru	Infuriated
గుర్రుగా	gurrugA	Horses
డిమాండ్	dimAMd	Demand
కఠిన	kaTina	Strict
నిర్బంధించారు	nirbaMXiMcAru	Detained
ఆగ్రహంతో	AgrahaMwo	Enraged
కోపం	kopaM	Anger
హెచ్చరిక	heVccarika	Warning
ఆరోపించారు	AropiMcAru	Accused

Table 7.8: Sample of sad emotion lexicons in Telugu, WX, and English

Telugu_word	WX_word	English_word
గాయపడ్డారు	gAyapaddAru	Injured
ఆందోళన	AMxolYana	Worry
అసంతృప్తి	asaMwqpwi	Discontent
హత్య	hawya	Murder
ఘర్షణ	GarRaNa	Clash
ఫ్లాప్లో	PlApwo	Flap
కుప్పకూలిన	kuppakUlina	Collapse
పరాభవం	parABavaM	Humiliation
పరాజయం	parAjayaM	Defeated
దారుణంగా	xAruNaMgA	Worse

Table 7.9: Sample of fear emotion lexicons in Telugu, WX, and English

Telugu_word	WX_word	English_word
హారర్	hArar	Horror
షాక్	kaTina	Shock
భయంతో	BayaMwo	Fear
హెచ్చరించి	heVccariMci	Alert
దడ	xada	Shore
భయపడి	Bayapadi	Worried
హడలిపోయారు	hadalipoyAru	Startled
హెచ్చరించడంతో	heVccariMcadaMwo	Warning
డెడ్	deVd	Dead
దాడి	xAdi	Attack

7.3.1.1 Building Telugu Lexicon Features

Lexicons play an important role when dealing with tasks like SA [50, 128, 39], EI [133, 83, 84], and HS [43, 15, 16]. Moreover, one can use these lexicons (e.g., positive and negative lexicons) as features to improve the accuracy of the models [61, 85, 23, 82]. These lexicons provide a word-level foundation for identifying polarity, emotion, or hatred in sentences and documents. This simple lexicon method shows surprising accuracy in [82].

In this chapter, we create sentiment ("positive", "negative" words), emotion ("anger", "sad", "happy", "fear" words), and hate-speech ("hate" words) lexicons manually from the annotated Telugu corpus. The detailed count of each task lexicons is shown in Table 7.3. To extract the sentiment/emotion/hate-speech lexicons, we ask the annotators to provide the main word that expresses the polarity/emotion/hate feeling and task labels. We show case the samples of "positive" and "negative" Telugu sentiment lexicons in Tables 7.4, and 7.5. Tables 7.6, 7.7, 7.8, 7.9

Table 7.10: Sample of hate-speech lexicons in Telugu, WX, and English

Telugu_word	WX_word	English_word
హింస	hiMsa	Violence
వార్నింగ్	vArniMg	Warning
ఆగ్రహం	AgrahaM	Wrath
నిప్పు పెట్టారు	nippu peVttAru	Set on fire
హెచ్చరించారు	heVccariMcAru	Warned
మండిపడ్డారు	maMdipaddAru	Infuriated
అసభ్యకరంగా	asaByakaraMgA	Obscenity
ఘర్షణ	GarRaNa	Friction
అవమానించే	avamAniMce	Demeans
విషవాయువుల	viRavAyuvula	Fumes

report the samples of “happy”, “sad”, “fear”, and “angry” Telugu emotion lexicons. We display the Telugu hate-speech sample lexicons in Table 7.10. Since sarcasm is very difficult to identify through a lexicon, as one person may feel that word/phrase sarcastic, and others may feel it real. Also, it became difficult to tell the annotators regarding this. So, we decided to create lexicons for the remaining three tasks (SA, ED, and HS).

7.3.1.2 Extraction of Telugu Lexicon Features

We built the lexicon feature representation of each sentence in the following way: 1) First, the lexicons are considered corresponding to their task (e.g., emotion lexicons used for emotion identification task). 2) For each sentence, we calculated the relative frequency of ”happy” words in a category of happy lexicons according to the Eq 7.1.

$$RF_h(W_c) = \frac{f(W_c)}{Total_h(W_c)} \tag{7.1}$$

Where, $f(W_c)$ represent the count of happy words in a sentence, $Total_h(W_c)$ denotes the total count of happy lexicons. 3) Similarly, we measured the relative frequency of “sad”, “fear”, and “anger” lexicons. 4) The final feature vector is the concatenation of all the individual features, i.e., dimension of four for the emotion task.

7.3.1.3 Telugu Bag-of-Words Representation

To extract the BoW features [126] for our data, we build a vocabulary from the annotated dataset. We filtered the stopwords (highly frequently occurring words in the Telugu language) and extracted the top 2000 words as our features resulted in a dataset size of 35,412 (samples) x 2000 (features). Since BoW provides the binary representation (word presence or absence), it can not capture semantics, grammar, word context (local and global), and order.

7.3.1.4 Telugu TF-IDF Representation

Some problems, such as the influence of high-frequency words, limit the good features when we train a BoW model. TF-IDF [105] is one such model that diminishes the problem of frequent high words. TF-IDF is mainly popular in the research community and industry due to its retrieval capability of documents based on ranking. TF-IDF suppresses the stop-words ranking automatically without the removal of stop-words in the training process. We use the top 2000 features for the training process. However, TF-IDF does not address semantics, grammar, and word context in a language model.

7.3.2 Generation of Telugu Word Embeddings

Word embedding methods allow the machine learning algorithms to understand and detect words with similar meanings. Moreover, these distributed word representations capture many

precise syntactic and semantic word relationships. This section describes the generation of different word embeddings such as Word2Vec-Te, GloVe-Te, FastText-Te, and Meta-Embeddings-Te for Telugu language and visualizes the semantic space captured across the words.

7.3.2.1 Word2Vec-Telugu (Word2Vec-Te)

Word2Vec model provides a non-deterministic way to determine the word representations [80]. It can learn similar word vectors for words in a similar context. Here, we generated the Word2Vec-Te embeddings i.e. Word2Vec on large Telugu corpora of 80,15,588 sentences. We used the Skip-Gram (SG) Word2vec [80] model with an extensive vocabulary size of 4,60,000 words where each word frequency is at least 5. The number of hidden neurons is 300, and the context window size is 5 as hyper-parameters for training the model.

7.3.2.2 GloVe-Telugu (Glove-Te)

The input used in the GloVe-Te model is a non-zero word-word co-occurrence matrix [95], which adds the global context information by default, unlike the use of local context in Word2Vec [80]. The corpora and training details are provided below:

- We tokenize the corpus with Polyglot tokenizer[2], build a vocabulary size of 4,60,000.
- The word vector dimension is 50.
- We use the parameters with a learning rate of 0.05, and the context window size is 10.

7.3.2.3 FastText-Telugu (FastText-Te)

Since the FastText model considers the bag of character n-grams to represent each word [56], it allows us to compute embedding representations for rare words. To learn the representations of a Telugu word, we follow the below steps in the training of FastText-Te.

- The word vector dimension is 200.
- We considered the words in the vocabulary with a frequency of at least 5 in the training set and obtained a vocabulary size of 4,62,232.
- We use the context window size as 5.

[2] `https://github.com/ltrc/polyglot-tokenizer`

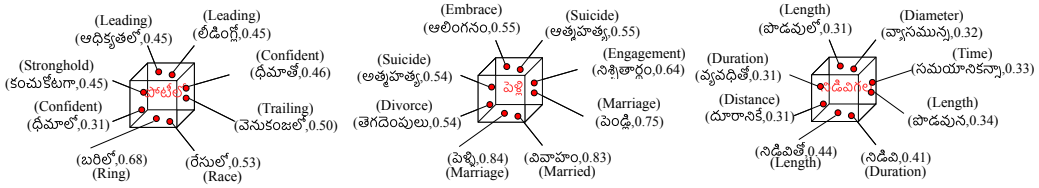


Figure 7.2: Top 8 similar words using Word2Vec-Te

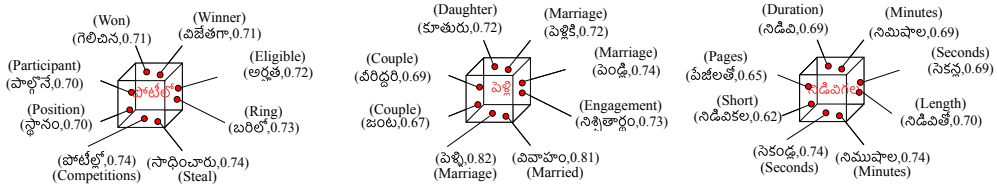


Figure 7.3: Top 8 similar words using GloVe-Te

7.3.2.4 Meta-Embeddings-Telugu (Meta-Embeddings-Te)

Meta-embeddings are shown to be successful for English, and it has two benefits compared to individual embedding sets: (i) enhancement of performance and (ii) improved coverage of the vocabulary [137]. These prior studies motivate us to create Meta-Embeddings for the Telugu language by using an ensemble of Word2Vec-Te, GloVe-Te, and FastText-Te embeddings. In this work, we created the Meta-Embeddings-Te using the average of encoded source embeddings. The dimension of Meta-Embeddings-Te is 300.

7.3.2.5 Qualitative Analysis of Telugu Word-Embeddings

Since word vectors are numerical representations of contextual similarities between words they can be manipulated and made to perform unusual tasks like finding the degree of similarity between two words, arithmetic operations among the words [80, 95]. To verify the quality of

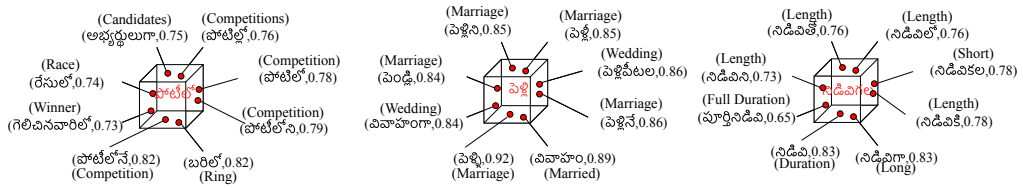


Figure 7.4: Top 8 similar words using FastText-Te

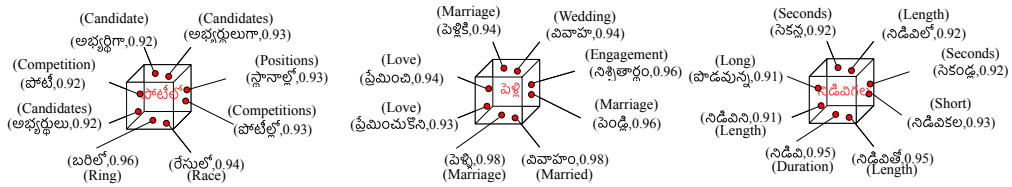


Figure 7.5: Top 8 similar words using Meta-Embeddings-Te

generated Telugu word-embeddings, we extract the top-8 semantic words for "(Competition) పోటీలో", "(Marriage) పెళ్లి", "(Short time) నిడివిగల". We obtained these results using the four generated word-embeddings mentioned above which are displayed in the Figures 7.2, 7.3, 7.4, and 7.5. In the Word2Vec-Te, the similar words are semantically similar to "(Competition) పోటీలో", "(Marriage) పెళ్లి", "(Short time) నిడివిగల". On the other hand, GloVe-Te method give related words along with semantically similar words like { (Winning) గెలిచినా, (winning) విజేతగా, (position) స్థానం } for "(Competition) పోటీలో", { (Daughter) కూతురు, (couple) జంట, (engagement) నిశ్చితార్థం } for "(Marriage) పెళ్లి", and {(minutes) నిమిషాల, (seconds) సెకండ్ల, (pages) పేజీలతో} for "(Short time) నిడివిగల".

FastText-Te feature vectors are related to morphological word representations and also address the problems like word semantics, context, and data sparsity [56]. From the Figure 7.4, we observe that words like { (competitions) పోటీల్లో, (competition) పోటీలోనే} for "(Competition) పోటీలో ", { (Bridesmaids) పెళ్లిని, (bridesmaids) పెళ్లిపీటలు } for "(Marriage) పెళ్లి", and {(Long) నిడివి, (Long) నిడివిని} for "(Short time) నిడివిగల" are related to morphological structure of the words. Moreover, we can observe that the semantic words from Figure 7.4 are similar to Word2Vec-Te and GloVe-Te. Finally, Meta-Embeddings-Te provides the advantages of three methods, Word2Vec-Te, GloVe-Te, and FastText-Te, with top similar words shown in Figure 7.5.

We performed arithmetic operations to evaluate the four-word embeddings. For example, as shown in Figure 7.6, we can retrieve words that have semantic relations from the Word2Vec-Te model, such as Organization-Company: vector(ఆపిల్) - vector(కంపెనీ) + vector(అరటి) = vector(బొప్పాయి). Similarly, words that have syntactic relationships such as the Singular-Plural: vector(మామయ్య) - vector(అతడు) + vector(అత్తా) = vector(భార్య). These empirical results show that the Word2Vec-Te approach successfully learned the meaning of words in the Telugu language. We also observe that FastText-Te and Meta-Embeddings-Te are better at performing the arithmetic operations similar to Word2Vec-Te. The first example in the GloVe-Te model resulted in successful arithmetic operations compared with the second example.

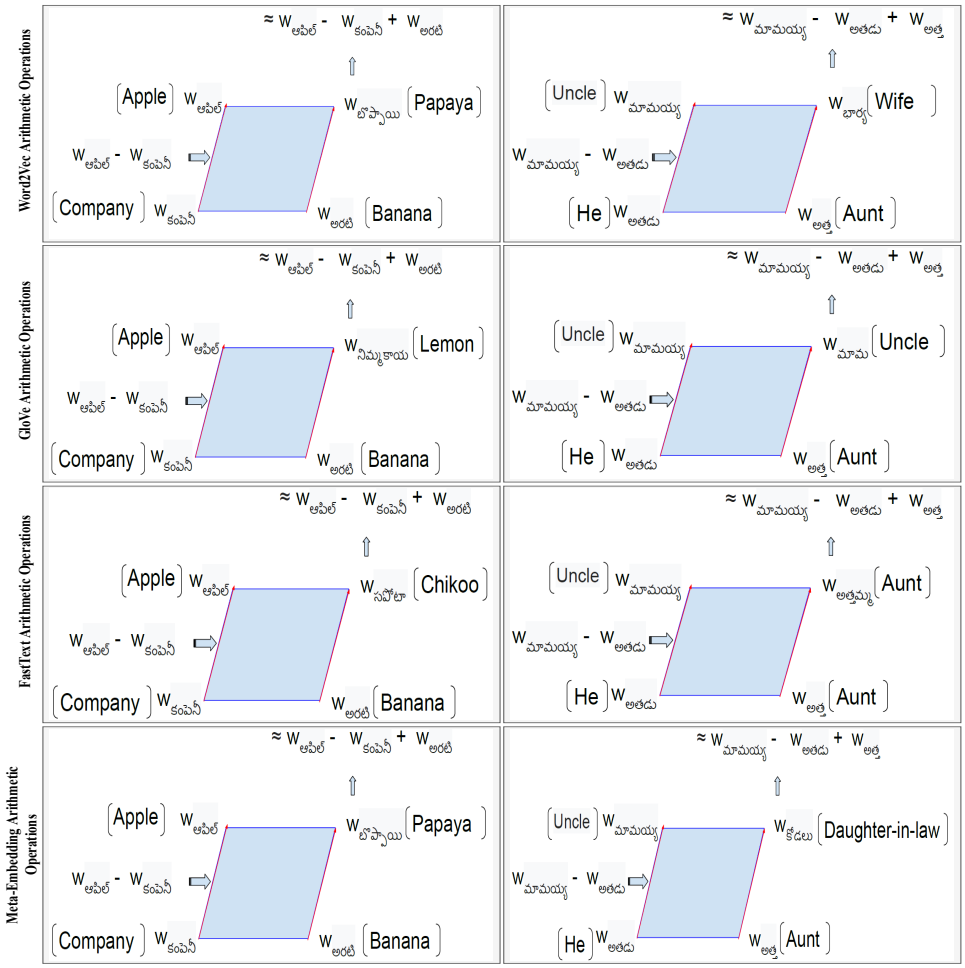


Figure 7.6: Word-Embedding Arithmetic Operations

7.3.3 Telugu Sentence Level Embeddings

In literature, several researchers use deep learning approaches for learning sentence vectors that mapped from word vectors such as recurrent networks [49], recursive networks [114], convolutional networks [62, 58], and recurrent convolutional networks [22, 146]. All these methods learn the representation of a sentence in a supervised fashion, i.e., depending on the class label. As a result, the representations of a sentence are of high quality but specific to the respective downstream task.

Motivated by the recent successful language models which try to learn the sentence vectors in an unsupervised fashion. This section discusses two methods, including Skip-Thought vectors and ELMo. We trained these two methods on Telugu corpus from scratch to produce the sentence vectors.

అమరావతి: రాష్ట్ర విభజన తర్వాత మోడీ కామెంట్లు కరెక్ట్ కాదని స్పీకర్ కోడెల శివప్రసాదరావు అన్నారు. Amaravathi: Speaker Kodela Sivaprasad Rao said that Modi's comments were not correct after the division of the state (s1).
తనను స్పీకర్ గా ఎన్నుకున్నందుకు అందరికీ ధన్యవాదాలు తెలియజేస్తున్నానని ఏపీ అసెంబ్లీ స్పీకర్ తమ్మినేని సీతారాం అన్నారు0. (0.692) AP Assembly Speaker Tammeneni Sitaram thanked everyone for making him a Speaker.
అమరావతి: శ్రీకాకుళం జిల్లా నుంచి ఎంపికైన నాలుగో స్పీకర్ తమ్మినేని సీతారాం అని వెల్లడి సభ్యులు ధర్మాన ప్రసాదరావు అన్నారు. (0.746) Amaravathi: Dharmamani Prasadara Rao said that the fourth speaker from the Srikakulam district is Tammeneni Sitaram.
విజయసాయిరెడ్డి మూర్ఖపు వెఖ్ఖరితో వారి పార్టీ నాయకులే విసిగెత్తారని ఎమ్మెల్సీ అశోక్ బాబు అన్నారు. (0.750) MLA Ashok Babu says that their party leaders themselves are fed up with the stupid attitude of Vijayasaray Reddy.
తమపై వచ్చిన ఆరోపణలపై ఒక్క ఆధారమైనా మూపించాలని ఈ సందర్భంగా కోడెల సవాల్ విసిరారు. (0.753) On this occasion, Kodela challenged to throw at least one evidence of the charges against them.
దీనిపై డీజీపీ ఠాకూర్ మాట్లాడుతూ:కిడ్నీ రాకెట్ కేసును సీపీ దర్యాప్తు చేస్తున్నారని అన్నారు. (0.773) DGP Thakur said: CP is investigating the Kidney Rocket case.
ఆంధ్రప్రదేశ్ ముఖ్యమంత్రి జగన్ వయస్సు చిన్నదైనా ఆయన బాధ్యత పెద్దదని తెలంగాణ ముఖ్యమంత్రి కేసీఆర్ అన్నారు. (0.776) Telangana Chief Minister KCR said that the responsibility of Andhra Pradesh Chief Minister Jagan is huge, though he is young.
మన పాలన దేశానికే ఆదర్శంగా ఉండాలని ముఖ్యమంత్రి జగన్మోహన్రెడ్డిఅన్నారు. (0.778) Chief Minister Jaganmohan Reddy said that their rule should stand ideal for the country.
ప్రతిపక్షంలో ఉన్న ఐదేళ్ల కాలంలో తాము విలువలు పాటించాము అని జగన్ అన్నారు. (0.781) Jagan said that they have maintained their values during the five years of opposition.

Figure 7.7: Top 8 nearest neighbour of sentences predicted using Skip-Thought-Te model for the input sentence s1 (top row)

7.3.3.1 Telugu Skip-Thought Representations (Skip-Thought-Te)

The Skip-Thought model is a sentence encoding-decoding model that produces a fixed-length vector for every given sentence [67]. Inspired by the Word2Vec approach, the Skip-Thought model extends the skip-gram model to generate sentence embeddings from word embeddings. The Skip-Thought model consists of three parts such as (i) an encoder network, (ii) a previous decoder network, and (iii) the following decoder network. We use the same Skip-Thought architecture that is RNN encoder with GRU and RNN decoder with conditional GRU to train the model [25]. The training process of the Skip-Thought model for Telugu corpus is as follows

- We use a mini-batch size of 32, Adam optimizer in training.
- Initially, we trained on 20,000 vocabulary from the Telugu corpus.
- We trained a uni-directional encoder (uni-skip) with 2400 dimensions.
- The number of epochs to train the model is 5,00,000.
- We expanded the vocabulary size of 4,60,000 using vocab expansion and skip-gram Word2Vec vectors.
- When encoding sentences, no other pre-processing is done other than tokenization using Polyglot tokenizer.
- The model runs for three days on a 16GB GPU machine.

ఈ అద్భుతమైన పని పంతొమ్మిదవ శతాబ్దం ప్రారంభంలో జరిగింది. I axBuwamEna pani paMwoVmmixava SawAbxaM prAraMBaMlo jarigiMxi. This amazing work was done in the early nineteenth century.
ఉద్యోగి ప్రతిరోజూ కష్టపడి పనిచేస్తాడు. uxyogi prawirojU kaRtapadi paniceswAdu. Employee works hard every day.
ఆండీ వార్హోల్ చేసిన ఈ అందమైన పని నాకు చాలా ఇష్టం. AMdI vArhol cesina I aMxamEna pani nAku cAlA iRtaM. I like this beautiful work by Andy warhol.
ఈ భవనంలో వందలాది మంది పనిచేస్తున్నారు. I BavanaMlo vaMxalAxi maMxi paniceswunnAru. Hundreds of people are working in this building.
నా సోదరి స్టార్‌బక్స్ వద్ద పనిచేస్తుంది. nA soxari stArbaks vaxxa paniceswuMxi. My sister is working at starbucks.

Figure 7.8: Five sentences used in ELMo-Te visualization

To evaluate our skip-thought-Te model, we predicted the top-8 nearest sentences for a given input query shown in Figure 7.7. These nearest neighbors were scored by cosine similarity from a random sample of 20,00,000 sentences from our corpus. From the Figure 7.7, we observe that the nearest neighbor sentences are related to their corresponding input query.

7.3.3.2 Telugu ELMo Embeddings (*ELMo-Te*)

Earlier word embeddings such as Word2Vec-Te, GloVe-Te, FastText-Te, and Meta-Embeddings-Te provide a unique word representation throughout our Telugu corpus. This unique representation is the main limitation of these methods, especially words with different contexts. To overcome these limitations and capture the long-term dependencies, researchers proposed a new architecture, Embeddings from Language Models (ELMo). ELMo is a successful NLP framework developed by AllenNLP [97]. Unlike earlier embeddings, these ELMo embeddings represent the words contextually using a bidirectional LSTM model. Moreover, ELMo embeddings do not miss the out of vocabulary words due to character embedding representations for each token and provide state-of-the-art results in many NLP tasks. This chapter creates *ELMo-Te* embeddings for Telugu, trained on large corpora of 8 million sentences. To train the *ELMo-Te* model from scratch for Telugu, we provide the data in the following format.

- First, we built a vocabulary file that contains all the unique words along with three extra tokens <S>, </S>, </UNK>.
- The vocabulary file is in descending order where the most frequently occurring word is at the top.
- The data file contains one sentence per line from the corpus.

- We split the data into different files in which 70% of the data is considered for training, while the remaining data is held out for testing.
- A configuration file consists of the vocabulary size, the number of tokens to train the ELMo model, and the output dimension of each token.
- We used 262 characters for training.
- Each embedding is of 1024 dimensions.
- Total number of train tokens is 2,086,488 and vocabulary size 793,384.

Figure 7.9: Layer-1 ELMo-Te vector for the word "పని/pani/work"

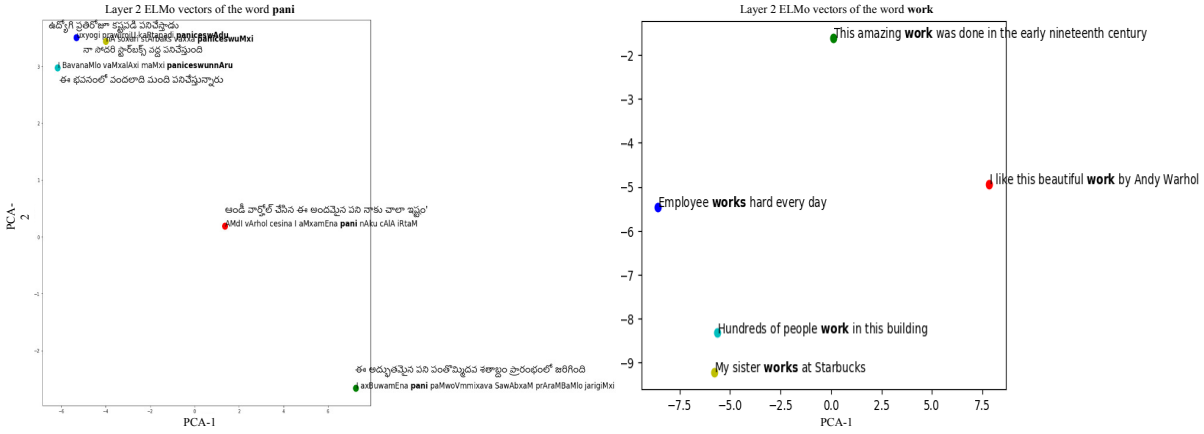


Figure 7.10: Layer-2 ELMo-Te vector for the word "పని/pani/work"

Visulazing *ELMo-Te* Contextual Vectors: ELMo embeddings are used to generate the context vectors, and for visualization, principal component analysis (PCA) is used to project

the vectors to a 2D space. We will visualize layer 1 and 2 contextual vectors for the word with multiple senses - "work". "Work" is a word that acts as a noun(means something done or made) and a verb(means to work). We considered the five sentences depicted in Figure 7.8 that contain either one of the two senses mentioned above: Figures 7.9, 7.10 are the projection results using PCA for the 5 sentences mentioned in Figure 7.8. Looking at layer one vectors shown in Figure 7.9, we can not immediately tell one cluster apart from the other. However, we know that the left three vectors have "work" as a verb, and the right two vectors have "work" as a noun. For layer two vectors from Figure 7.10, we observe apparent clustering of work as a verb in the lower right corner and then another cluster of work as a noun in the upper left corner. The above results indicate that the cluster formations seem more precise, and the distance between cluster centroid is larger in the layer-2 *ELMo-Te* vector space than in layer-1. Moreover, layer-2 handles the Word Sense Disambiguation task with a higher F1-score than using the first layer [46]. Our observation of the greater distance in the layer-2 vector space from Figure 7.10 shows a possible explanation of similar finding for the *ELMo-Te* embeddings as well.

7.3.4 Telugu Transformer Models

7.3.4.1 BERT-Telugu (*BERT-Te*):

BERT is a pre-trained language model [35] that provides bi-directional contextual information, while earlier methods have uni-directional context. Like pre-trained BERT [35] (model trained on the Books Corpus [150] and English Wikipedia), we choose a model based on the transformer structure of BERT-base-cased for Telugu (large corpora of 8 million sentences) [78]. The BERT-base-cased model consists of 12 transformer blocks, 768 hidden dimensions, 12 self-attention blocks, and 110 million parameters in total.

7.3.4.2 RoBERTa-Telugu (*RoBERTa-Te*):

The recent successful RoBERTa model [75] is an optimized pre-trained language model that improves on BERT, trained on 160GB of text (3-4 times the amount of data used for pretraining BERT). Similar to *BERT-Te*, we build *RoBERTa-Te*, a pre-trained RoBERTa-base model for Telugu (large corpora of 8 million sentences). The RoBERTa-base model consists of 12 transformer blocks, 768 hidden dimensions, 12 self-attention blocks, and 110 million parameters in total.

7.3.4.3 ALBERT-Telugu (*ALBERT-Te*):

ALBERT (a lite BERT) [71] model is a novel pretraining method that mainly contributed to three solutions to overcome limitations of the BERT model, including (i) parameter reduction

using factorized embedding parameterization, (ii) cross-layer parameter sharing, and (iii) inter-sentence coherence loss. To create a Telugu pre-trained ALBERT model (*ALBERT-Te*) for Telugu, we use an ALBERT-base-v2 model consisting of 12 transformer blocks, 128 hidden dimensions, 12 self-attention blocks, and 11 million parameters in total.

7.3.4.4 DistilBERT-Telugu (*DistilBERT-Te*):

DistilBERT [110] is a reduced BERT's model that shows the effectiveness of transformer-based language models in a production setting as well as reduces BERT's model size by 40%. The *DistilBERT-Te*) Model is pre-trained on 8 million Telugu sentences where the model consists of only 66 million parameters.

7.4 Experimental Setup and Methods Training

To measure the performance of four tasks, we perform the following experiments: (i) First, we create the baseline results using lexicons, BOW, and TF-IDF. (ii) Second, we use the Telugu pretrained word embedding methods such as Word2Vec-Te, GloVe-Te, FastText-Te, and Meta-Embeddings-Te to compare the performance with the baseline models for four of the NLP tasks. (iii) Third, we create models on sentence specific embeddings, Skip-Thought-Te vectors, and context-specific embeddings *ELMo-Te*. (iv) Finally, we use Telugu pretrained transformer models such as *BERT-Te*, *RoBERTa-Te*, *ALBERT-Te*, and *DistilBERT-Te* for overall comparison. We use two classifier methods in the training process, including simple Logistic Regression (LR) [138], and a deep learning method LSTM [49]. To understand how each class performs, we choose classification metrics such as macro-average precision, recall, and F1-score that give equal weight to each class to evaluate system performance.

7.4.1 Logistic Regression Training

The input to the logistic regression model [138] is a text vector extracted from one of the feature representation methods mentioned above, and the target output consists of class labels depending on the task we used in training. Formally, we represent the extracted text vector as $X \in \mathbb{R}^{N \times D}$ and target class labels $Y \in \mathbb{R}^{N \times C}$, where N denotes the number of training examples, D denotes the dimension of input text representation, and C denotes the number of classes in a task. The logistic regression objective function for the i^{th} example is given as follows.

$$f(\theta) = \frac{1}{N}[\sum_{i=1}^{N} -Y_i \log(h_\theta(X_i)) + (1 - Y_i) \log(1 - h_\theta(X_i))]$$

Here, θ are the learnable weight parameters.

We consider the LR results in our baseline training results to compare with the linear probing and fine-tuning the transformer model results. In the multi-class case, we set to "multinomial" while training the ED task with Logistic regression. We train a Logistic Regression classifier on training data with regularization by passing the parameters: {C = 20.0, dual = False, penalty = l2}.

7.4.2 Imbalanced Dataset Handling

From Table 7.2, we observe that each task data has a class imbalance issue; for example, from Table 7.2, we observe that the majority class ("no") has ∼10x more instances than the "yes" class in hate-speech and sarcasm tasks. To circumvent this problem, we tried methods such as SMOTE (Synthetic Minority Oversampling Technique) [19] based over-sampling and random under-sampling [135] techniques.

7.4.3 LSTM Method Training

In the literature, LSTMs [49] are successful in dealing with sequence-based problems in the field of NLP [148, 141]. Using LSTM architecture, we tried the five different feature representations as input in the training process. Rather than averaging each word vector to get the sentence representation, we learn sentence embeddings through an LSTM model. We train the LSTM model as follows: (i) Sequence of tokens with embedding features as input at the input layer, and (ii) Sequence of tokens with four Telugu pretrained word embeddings (Word2Vec-Te, GloVe-Te, FastText-Te, and Meta-Embeddings-Te) features as token input at the input layer. Further, we add two dense layers with 256 and 128 neurons. Finally, we predict the class label for each NLP task at the last step.

7.4.4 Challenging Tasks

Hate-speech and Sarcasm prove to be the more challenging tasks. On both tasks, Telugu pretrained language models improve on *mBERT*, *XLM-R*, and *IndicBERT*.

7.4.5 Fine-tuning of Pretrained Transformers

In order to fine-tune a pretrained Transformer model, we observe that the following hyper-parameters yields best performances: (i) Batch size: *32*, (ii) Learning rate: $3e^{-5}$, (iii) Number of training epochs: *10*, (iv) ϵ constant set to $1e^{-8}$ to avoid division by zero in the AdamW calculation when the gradient approaches to zero, and (v) *AdamW* as optimizer. To avoid the over-fitting problem, we stopped training if the validation loss did not decrease for five consecutive epochs. To account for memory constraints, we use a smaller maximum sequence length of 128 during fine-tuning of pretrained models.

7.4.6 Dataset Splitting

We split our labeled dataset into train/dev/test (70%/10%/20%), where we fine-tuned our models with train & dev datasets and results on a held-out test set used for only the best system.

7.4.7 Model Training Details

We train and fine-tune all our models on a single GPU (11GB RAM) with Nvidia GEForce GTX.

7.5 Telugu Pretrained Word and Sentence Embeddings

7.5.1 Text Classification Evaluation

Here, our main objective is to perform text classification of four NLP tasks: (i) SA: the task is to predict the sentiment class ("positive", "negative", "neutral") of a given sentence, (ii) EI: the task is to predict the emotion class ("Neutral", "Angry", "Fear", "Happy", and "Sad"), (iii) HS: the task is to predict the hate-speech class ("Yes" or "No"), and (iv) SAR: the task is to predict the given sentence has sarcasm or not. To handle the class-imbalance problem, we use the under-sampling [135] & over-sampling [19] methods to calculate the results. Here, we present the results of SA, EI, HS, and SAR in Tables 7.11(a), 7.11(b), 7.12(a), and 7.12(b), respectively.

Table 7.11: Text Classification Results using Different feature sets with Logistic Regression classifier under different sampling methods with No/Over/Under-Sampling: (a) SA (b) EI.

	(a) Sentiment Analysis								
Sampling Method→	NS			OS			US		
Feature set↓	P	R	F1	P	R	F1	P	R	F1
Lexicon-Based	49	63	51	59	56	57	54	53	54
BOW	58	51	52	53	55	54	50	53	50
TF-IDF	60	51	52	55	57	55	52	55	52
Word2Vec-Te	62	54	56	58	62	59	56	60	57
GloVe-Te	61	52	54	58	62	58	55	59	55
FastText-Te	64	57	59	**62**	**67**	**63**	60	64	61
Meta-Embeddings-Te	64	51	53	60	64	61	57	61	58
Skip-Thought-Te	60	54	55	60	57	58	54	59	55
ELMo-Te	65	61	62	**62**	**66**	**63**	61	65	62

P = Precision, R = Recall, F1 = F1-score

	(b) Emotion Identification								
Sampling Method→	NS			OS			US		
Feature set↓	P	R	F1	P	R	F1	P	R	F1
Lexicon-Based	40	27	27	30	35	30	29	34	28
BOW	44	30	31	32	36	32	24	35	19
TF-IDF	53	28	31	32	42	32	24	33	20
Word2Vec-Te	50	29	32	33	30	33	30	49	28
GloVe-Te	43	27	29	32	49	30	30	45	26
FastText-Te	50	32	35	35	57	36	29	49	27
Meta-Embeddings-Te	42	25	25	34	54	34	30	47	25
Skip-Thought-Te	52	30	33	33	44	35	28	44	24
ELMo-Te	55	37	39	**38**	**54**	**41**	31	50	29

P = Precision, R = Recall, F1 = F1-score

SA: From Table 7.11(a), we observe that lexicon-based sentiment features as input in the over-sampling setup report an F1-score of 57% better than BoW (54% F1-score) and TF-IDF (55% F1-score). We observe an increase in recall for standard features using under-sampling methods compared with the lexicon approach. Among the four Telugu pretrained word-embeddings, FastText-Te, and Meta-Embeddings-Te features improve the F1-score by 6-8% compared to baseline. For ELMo-Te features, we observe an F1-score of 63% similar to FastText-Te and

Table 7.12: Text Classification Results using Different feature sets with Logistic Regression classifier under different sampling methods with No/Over/Under-Sampling: (a) HS (b) SAR.

	(a) Hate-Speech								
Sampling Method→	NS			OS			US		
Feature set↓	P	R	F1	P	R	F1	P	R	F1
Lexicon-Based	50	49	50	55	60	56	56	57	56
BOW	49	50	50	52	52	50	51	69	45
TF-IDF	49	50	50	52	52	50	51	71	47
Word2Vec-Te	87	52	54	55	78	54	52	79	48
GloVe-Te	66	51	52	54	82	51	51	75	46
FastText-Te	92	55	58	55	80	54	52	81	50
Meta-Embeddings-Te	50	50	50	55	81	53	52	78	49
Skip-Thought-Te	50	50	50	55	71	58	52	77	48
ELMo-Te	67	59	60	**58**	**66**	**61**	52	82	51

P = Precision, R = Recall, F1 = F1-score

	(b) Sarcasm								
Sampling Method→	NS			OS			US		
Feature set↓	P	R	F1	P	R	F1	P	R	F1
BOW	49	50	50	50	51	48	50	50	49
TF-IDF	50	50	50	51	54	49	50	60	39
Word2Vec-Te	49	50	50	50	54	44	50	57	37
GloVe-Te	49	50	50	50	58	41	50	53	36
FastText-Te	49	50	50	51	55	42	50	58	39
Meta-Embeddings-Te	49	50	50	50	58	42	50	57	36
Skip-Thought-Te	50	50	50	53	63	53	51	73	46
ELMo-Te	58	52	53	**56**	**64**	**58**	52	76	46

P = Precision, R = Recall, F1 = F1-score

Meta-Embeddings-Te. Statistical significance using one-Anova test [60] on the over-sampling results (F1-score) from Table 7.11(a) provides an F-statistic [F(8,20) = 24.34,P = 6.9e-9] which concludes that nine feature representations are significantly different. We use the post-hoc Tukey-HSD test [109] to obtain the results between different pairs such as ELMo-Te vs. GloVe-Te (0.3782), BoW vs. Sentiment-lexicons (0.9), BoW vs. TF-IDF (0.7704), and BoW vs. Skip-Thought (0.0447), all these pairs are not significantly different. On the other-hand, the statistically significant different pairs are (i) Sentiment-lexicons vs Meta-Embeddings-Te (0.001), (ii) Sentiment-lexicons vs Word2Vec-Te (0.0034), (iii) Meta-Embeddings-Te vs TF-IDF (0.001), and (iv) TF-IDF vs Word2Vec-Te (0.0098).

EI: Table 7.11(b) list the results of five-class EI task. The first block describes the baseline feature results, where we use the emotion lexicons and traditional feature methods like BoW and TF-IDF as input. The emotion lexicon features as input yield an F1-score of 30% is lower than traditional feature representations BoW (32% F1-score) and TF-IDF (32% F1-score). The second block displays the four Telugu pretrained word embedding feature results, where FastText-Te outperforms the baseline with an F1-score of 36% in the over-sampling strategy. Moreover, the remaining pretrained word embeddings except GloVe-Te perform better than traditional methods. These results show that the pretrained Telugu word embeddings can capture semantic information. The last block report the results of sentence-specific embeddings and contextual embeddings. ELMo-Te outperforms both baseline and Telugu pretrained word embeddings with an F1-score of 41%. In contrast, the skip-thought-Te achieves an F1-score of 35% similar to word-embedding results.

We use F1-scores of each feature representation over-sampling results to perform the Anova test. The one-way ANOVA test provides an F-statistic [F(8,20) = 5.058,P = 0.00124] concluding that the nine feature representations are significantly different. We use the post-hoc Tukey-HSD test to obtain the results between various pairs such as ELMo-Te vs. GloVe-Te (0.1131), BoW vs. Emotion-lexicons (0.9), BoW vs. Tf-IDF (0.9), BoW vs. Skip-Thought-Te (0.8331) and state that they are not significantly different. On the other hand, the statistically significant different pairs are (i) BOW vs ELMo-Te (0.0077), (ii) ELMo-Te vs Emotion-lexicons (0.0025). These

results indicate that FastText-Te, Meta-Embeddings-Te, and ELMo-Te feature methods are statistically significant.

HS: For the hate-speech task, we report the results in Table 7.12(a). The first three rows in Table 7.12(a) depict the baseline results. Observations from Table 7.12(a) that hate-speech lexicons have an improved F1-score of 56% compared to traditional features BoW (51%) and TF-IDF (53%). The four Telugu pretrained word-embedding features show equally better results in the over-sampling setting. Also, these four embeddings resulted in a good recall score in the under-sampling setting. ELMo-Te outperforms the word-embeddings and baseline features among the contextual embeddings with an F1-score of 61% and a recall of 66% in the over-sampling setting. Although skip-thought features do not perform well on SA and EI tasks, having an increased F1-score of 58% and recall of 71% in the HS task. Moreover, skip-thought-Te features outperform the baseline and pretrained word embedding features.

The one-way ANOVA test provides an F-statistic [$F(8,20) = 5.134, P = 0.00113$] which concludes that the nine feature representations are significantly different. We use the post-hoc Tukey-HSD test to obtain the results between various pairs such as ELMo-Te vs. GloVe-Te (0.5318), BoW vs. hate-speech lexicons(0.1657), BoW vs. Tf-IDF (0.9), BoW vs. Skip-Thought-Te (0.0792) and state that they are not significantly different. On the other hand, the statistically significant different pairs are (i) BOW vs ELMo-Te (0.0015), (ii) ELMo-Te vs Tf-IDF (0.0015).

SAR: Among the four NLP tasks we used in this paper, sarcasm is challenging as it requires a deep understanding of contextual differences in a sentence. Despite the above challenge, our feature methods perform well on SAR Telugu data, as shown in Table 7.12(b). The baseline features resulted as BoW (F1-score 49%), TF-IDF (F1-score 51%), and word-embedding features as input perform extremely low and equal to random guessing. Also, the performance is low compared to the other three NLP tasks mentioned above. The last block displays the sentence-specific and contextual embeddings, where the ELMo-Te feature method outperforms all the baseline and the other embedding methods with an F1-score of 58%.

The one-way ANOVA test provides an F-statistic [$F(7,18) = 2.28, P = 0.069$] which concludes that eight feature representations are not significantly different. We use the post-hoc Tukey-HSD test to obtain the results between different pairs such as ELMo-Te vs. GloVe-Te (0.1087), BoW vs. Tf-IDF (0.9), BoW vs. Skip-Thought-Te (0.6872), which states that they are not significantly different.

From the Table 7.11(a), 7.11(b), 7.12(a), and 7.12(b), we make the following insights:

- Contextualized word-embedding *ELMo-Te* performs better in solving the identification of SA, EI, HS, and SAR even in a resource-poor language setting.
- Skip-thought-Te features are lacking in dealing with the sentiment labels.
- Skip-thought-Te features achieved better F1-score in EI task.

- The word embeddings without context representation as input perform equally with contextual word embedding *ELMo-Te*.
- The use of sentiment and emotion lexicons achieves better performance than Bow and TF-IDF.
- The pretrained word embeddings with n-gram representation (FastText-Te) better categorize the hate-speech samples.
- The use of under-sampling and over-sampling methods yield better recall and F1-score in the imbalanced setting.

7.5.2 Error Analysis

We now provide a detailed error analysis of the four NLP tasks, and we report the confusion matrices using the *ELMo-Te*.

SA: The confusion matrices for SA are depicted in Table 7.13. We observe from Table 7.13 that F1-score of the (i) "Negative" class is (63%), (ii) "Neutral" class is (69%), and (iii) "Positive" class is (57%) showcase the major difference. In ELMo-Te feature representation, the "positive" class performance is better, whereas the "negative" class results are comparatively low. Among the 3-classes, our system confused between "neutral" vs. "positive" classes and "neutral" vs. "negative" classes. We also perform the qualitative analysis of the predictions as presented next. Our system makes incorrect predictions where the sentences consist of the implicit sentiment. In the following sentence, the aspect term is "హిట్| hit" denoting "positive" sentiment. However, the system predicts the target label as "negative" sentiment as captured from the phrase "పెద్ద హిట్ |huge hit".

- Telugu Sentence: ఆల్ రెడీ పాటలు చాలా పెద్ద హిట్ అయ్యాయి.
- Wx Sentence: Al reVdI pAtalu cAlA peVxxa hit ayyAyi.
- English: Already, songs have been a huge hit.

EI: The confusion matrix for the EI is depicted in Table 7.14. We observe from the Table 7.14 that F1-score of the (i) "Anger"(29%), (ii) "Fear"(9%), (iii) "Happy"(82%), (iv) "Neutal"(79%), and (v) "Sad" class (54%) is the performance difference between five classes. In ELMo-Te feature representation, classes such as "Happy", "Neutral", and "Sad" have better performance than the remaining two classes. This performance difference is due to the fewer samples in "Anger" and "Fear" classes. Among the five-classes, our system confused between "Neutral" vs. "Happy", "Neutal" vs. "Sad", and "Happy" vs. "Sad" classes. Our system makes incorrect predictions where the sentences consist of implicit emotion. In the following sentence, the actual emotion is "Anger". However, the system predicts the target label as a "Sad" emotion due to the presence of the word "డెత్ death".

- Telugu Sentence: నకిలీ డెత్ సర్టిఫికెట్లతో రూ:కోట్లు టూఠీ చేసినట్లు తెలుస్తోంది.
- Wx Sentence: nakilI deVw sartiPikeVtlawo rU:kotlu lUtI cesinatlu weVluswoMxi.
- English: It is reported that crores of rupees have been looted with fake death certificates.

HS: We now describe the detailed error analysis of the hate-speech detection and report the confusion matrix in Table 7.15. From Table 7.15, we observe that the F1-score of the (i)"No" class yields 99% (ii)"Yes" class reports 30% is the main difference between the performance of the two classes. In ELMo-Te feature representation, the "No" class performance is better than the "yes" class due to class imbalance. However, ELMo-Te displays better performance when capturing hate-speech related content amidst feature representations. By considering the weighted average, our system displays an F1-score of 99%. We also perform the qualitative analysis of the predictions as presented next. The hate-speech model with ELMo-Te representation displays valid predictions for the sentences where most annotators are given incorrect labels. For example, the below sentence does not have the hate-speech word, but the annotators were given a 'Yes' label. However, our model correctly predicts the target label as "No", capturing the context.

- Telugu Sentence: తాజాగా మరోసారి ఈ భామల వ్యవహారం చర్చనీయాంశంగా మారింది.
- Wx Sentence: wAjAgA marosAri I BAmala vyavahAraM carcanIyAMSaMgA mAriMxi.
- English: The affair has become the subject of debate once again.

Our system makes incorrect predictions where the sentences consist of abusing words. In the following sentence, for the hate-speech terms "హెచ్చరించార| warning ", "ప్రాణాల దక్కవని |no Lives ", the actual label is "yes". However, the system predicts the target label as "no" for the input sentence.

- Telugu Sentence: మరోసారి ఇక్కడ కనిపిస్తే ప్రాణాలు దక్కవని కార్మికులు, నిర్మాణ సిబ్బందిని హెచ్చరించారు.
- Wx Sentence: marosAri ikkada kanipiswe prANAlu xakkavani kArmikulu, nirmANa sibbaMxini heVccariMcAru.
- English: The workers have warned the construction crew that their lives will not be found if they appear again.

SAR: We report the detailed error analysis of the sarcasm detection and report the confusion matrix in Table 7.16. From Table 7.16, we observe that the F1-score of the (i)"No" class yields 98% and (ii)"Yes" class reports 60% is the main difference between the performance of the two classes. In ELMo-Te feature representation, the "No" class performance is better than the "Yes" class due to class imbalance. However, ELMo-Te performs better when capturing sarcasm related content amidst feature representations. We also perform the qualitative analysis

Table 7.13: Confusion Matrix for SA with *ELMo-Te*

		Predicted		
		Negative	Neutral	Positive
Actual	Negative	1245	328	147
	Neutral	737	3020	968
	Positive	263	705	1373

Table 7.14: Confusion Matrix for EI with *ELMo-Te*

		Predicted				
		Anger	Fear	Happy	Neutral	Sad
Actual	Anger	61	0	0	19	18
	Fear	7	4	1	8	11
	Happy	5	2	572	361	88
	Neutral	136	28	863	4519	814
	Sad	70	4	75	281	839

Table 7.15: Confusion Matrix for HS with *ELMo-Te*

		Predicted	
		No	Yes
Actual	No	8647	40
	Yes	74	25

Table 7.16: Confusion Matrix for SAR with *ELMo-Te*

		Predicted	
		No	Yes
Actual	No	6685	34
	Yes	162	148

of the predictions as presented next. The sarcasm model with ELMo-Te representation displays valid predictions for the sentences where the annotation was complex. For example, the below sentence does not have the presence of idioms or exclamation marks. However, our model correctly predicts the target label as "Yes", capturing the context.

- Telugu Sentence: సెకండాఫ్‌లో మన బుర్రలకు పదును పెట్టేలా కొన్ని సీన్లు ఉండబోతున్నాయి.
- Wx Sentence: seVkaMdAPlo mana burralaku paxunu peVttelA koVnni sInlu uMdabowunnAyi.
- English: There are going to be some scenes in the second half to sharpen my hair.

Our model makes incorrect predictions where the sentences consist of "metonymy" in the SAR task. In the following sentence, the actual label is "Yes". However, the system predicts the target label as "No" for the input sentence.

- Telugu Sentence: న్యూజిలాండ్ బౌలర్లు తొలి మ్యాచ్‌లో శ్రీలంకను 136 పరుగుల తేడా తో చిత్తు చేశారు.
- Wx Sentence: nyUjilAMd bOlarlu woVli myAclo SrIlaMkanu 136 paru gulake kuppa kUlcAru.
- English: New Zealand bowlers thrashed Srilanka by 136 runs in the first match..

7.6 LSTM Model Results:

Figure 7.11 showcases both SA (left) and EI (right) results, where we train LSTM on (i) sequence of tokens and (ii) with the four-word embeddings. Here, we train the LSTM on five input representations, including (i) sequence of tokens, (ii) pretrained word embedding representations (Word2Vec-Te, GloVe-Te, FastText-Te, and Meta-Embeddings-Te) for the two

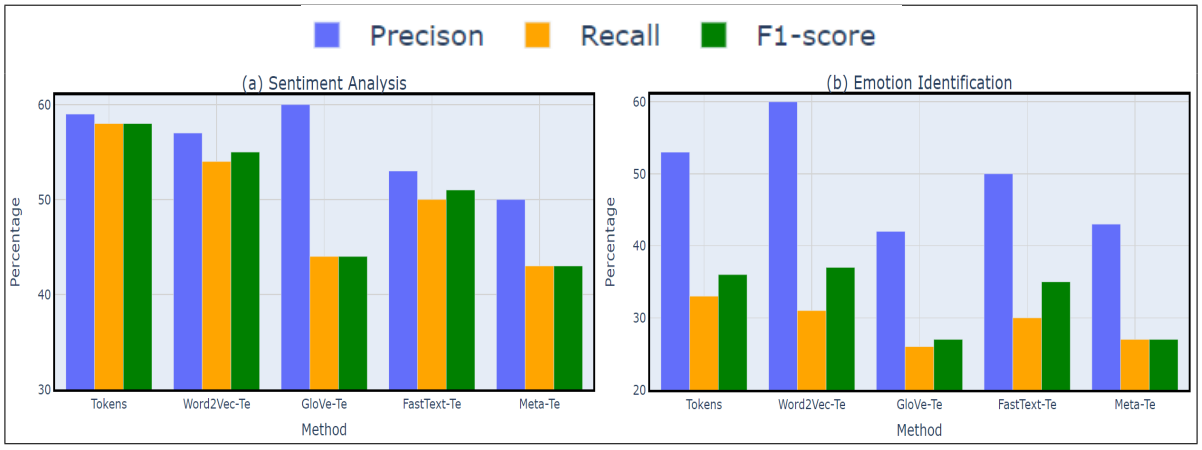


Figure 7.11: Comparison of F1-score performance of input word representations i) Sequence of Tokens ii) Word2Vec-Te iii) GloVe-Te iv) FastText-Te and v) Meta-Embeddings-Te. The LSTM model is used as a training model for each of the feature representations in the (a) SA and (b) EI setups.

NLP tasks: SA and EI. Here, we do not use the LSTM method for HS, and SAR tasks as these tasks have high class imbalance problems. When we train LSTM with a sequence of tokens in the first case, we consider the following parameters in training: For the first hidden layer, an embedding layer is used with 3-inputs (vocabulary size:15000, size of the vector space in which words embedded:512, length of input sequence:50); Our second hidden layer is an LSTM with 256 neurons as output; The last dense layer denotes the number of classes (3 for SA, 5 for EI); We use the Adam optimizer, and dropout is 0.5. The difference between the first and second cases is that we consider the pretrained word embedding feature dimension instead of vector space where words are embedded. From Figure 7.11, we observe that LSTM with the sequence of tokens as input yields a better F1-score of 58% and 37% for SA and EI, respectively. However, using pretrained word embedding vectors as input, the results of SA and EI tasks using LSTM are not better than the sequence of tokens. We assume that the LSTM model explores more during training and is thus more robust to learn word context at inference rather than pretrained word representations at embedding layer with no word context. Overall, the F1-measure of SA and EI for LSTM is lower than the results obtained from pretrained word and sentence embeddings, as displayed in Tables 7.11(a), 7.11(b), 7.12(a), and 7.12(b).

7.7 Comparison of Telugu Pretrained Transformers with *mBERT*, *XLM-R*, and *IndicBERT*

Table 7.17 report the training corpus size and model parameters of Telugu pretrained Transformers as well as multilingual pretrained Transformers such as *mBERT* [34], *XLM-R* [70], and

IndicBERT [57]. Here, we noticed that both *mBERT* and *XLM-R* are pretrained on Wikipedia data of 104 languages whereas *IndicBERT* is pretrained on Indian languages corpora (OSCAR) [117] (English corpora translated to each Indian language). On the other hand, Telugu pretrained Transformers are trained on Telugu Wikipedia data (1.6 million sentences) and the corpora extracted from many popular Telugu websites (6.4 million sentences).
We now introduce both linear probing and fine-tuning of Telugu pretrained Transformers and compare with multilingual pretrained Transformers. We perform linear probing and fine-tuning on four NLP tasks (SA, EI, HS, and SAR) and the available state-of-the-art Telugu datasets. We address the following challenges:

7.7.1 Do Linear Probing on Telugu Pretrained Transformers Outperform the existing Multilingual Language Models?

In this subsection, we present the comparative results on linear Probing of Telugu pretrained and existing Multilingual Transformers, as reported in Table 7.18. For four NLP tasks, we extract the [CLS] token representation for each input sentence using Telugu pretrained transformers as well as existing multilingual language models: mBERT [34], Cross-Lingual Language Model (XLM) [70], and IndicBERT [57]. Further, we compare the existing pretrained word embeddings such as Sub-word Byte Pair embeddings [48], and Facebook FastText word-embeddings [13]. We observe that probing on Telugu pretrained Transformers including, *BERT-Te*, *RoBERTa-Te*, and *DistilBERT-Te* outperform the existing pretrained embeddings across three NLP tasks: SA, EI, and SAR. In the two-class setup for the hate-speech task, the performance of all the models are similar to random guessing. This is mainly due to the presence of a high class imbalance problem. Overall, these results indicate that the performance of probing results from Table 7.18 are lower than results in Table 7.11(a), 7.11(b), and 7.12(a) except for the SAR task.

Table 7.17: Models and their Training Corpus sizes. B: Billions, M: Millions

Model	#Sentences	#Parameters
mBERT	18.2B (Telugu 1.6M)	110M
XLM-R	295B (Telugu 1.6M)	125M
IndicBERT	452.8M (Telugu 47.9M)	12M
BERT-Te	8.1M	108M
RoBERTa-Te	8.1M	125M
DistilBERT-Te	8.1M	66M
ALBERT-Te	8.1M	11M

7.7.2 Do Fine-tuning of Telugu Pretrained Transformers Outperform the Existing Multilingual Language Models?

After linear probing on pretrained models, we evaluate whether fine-tuning the Telugu pretrained and existing multilingual transformer models performs better than probing tasks for

Table 7.18: Linear Probing Results: Comparison of Text classification results with linear probing on existing pretrained Telugu word embeddings and our Telugu pretrained Transformers on four different NLP tasks.

	Sentiment	Emotion	Hate-Speech	Sarcasm
Feature set↓	F1-score	F1-score	F1-score	F1-score
Facebook Fast Text	40.03	21.13	49.90	49.20
Subword BYte Pair	40.10	18.07	50.00	49.60
mBERT	40.03	23.21	49.50	49.30
XLM-R base	38.20	19.54	49.60	49.80
IndicBERT	40.68	30.16	50.00	66.49
BERT-Te	**49.67**	43.74	**52.61**	**68.59**
ALBERT-Te	41.25	28.16	49.90	61.32
RoBERTa-Te	45.86	**44.21**	50.00	66.74
DistilBERT-Te	43.45	38.53	50.00	66.49

each of the four NLP tasks. In order to fine-tune the pretrained transformer, we use Simple-Transformers library [3], and the hyper-parameters used for fine-tuning is mentioned in subsection 7.4.5. From Table 7.19, we find that fine-tuning results show better performance than baseline, pretrained word embeddings, sentence embeddings, and probing methods. From Table 7.19, we observe that both *BERT-Te* and *RoBERTa-Te* models yield higher F1-scores than other models for four NLP tasks.

Statistical Significance In order to estimate the statistical significance of the performance differences, we performed the *post hoc* pairwise tests for all the models across the four NLP tasks. We found that *BERT-Te* is statistically significant and better than *mBERT*, *XLM-R*, and *IndicBERT* for all tasks except for SAR. Lastly, *IndicBERT* is not statistically significant with *mBERT* and *XLM-R* for three NLP tasks such as SA, EI, and SAR except for HS task. Further, we compared *BERT-Te* with *RoBERTa-Te*, and *DistilBERT-Te* models. From Table 7.20, we observe that *BERT-Te* is not statistically significant with *RoBERTa-Te*, and *DistilBERT-Te* models for SA, EI, and SAR except for HS task. Detailed p-values are mentioned in Table 7.20, the $*$ denotes that the two pairs of models are significantly different (i.e. p 0.05).

AUC-ROC Curves To evaluate the model performance of two best performing representations (*BERT-Te*, and *RoBERTa-Te*), we plot the Area Under Curve Receiver Operator Characteristic, a.k.a AUC-ROC curves that show the performance of a classification model at all possible thresholds. Figures 7.12, 7.13, 7.14, and 7.15 report the ROC plots for the four NLP tasks. These ROC curves typically feature a false positive rate on the X-axis and a true positive rate on the Y-axis. It is evident from the Figures 7.12, 7.13, 7.14, and 7.15 that the AUC-ROC cruve for the Logistic Regression is doing a better a job across each class for the four NLP tasks.

[3]https://simpletransformers.ai/

Table 7.19: Fine-tuning Results: Comparison of Text classification results with fine-tuning on existing multilingual pretrained Transformers as well as our Telugu pretrained Transformers on four different NLP tasks.

	Sentiment	Emotion	Hate-Speech	Sarcasm
Feature set↓	F1-score	F1-score	F1-score	F1-score
BERT-Te	**68.72**	56.66	**64.27**	76.28
ALBERT-Te	63.82	54.95	58.23	75.84
RoBERTa-Te	66.09	**58.04**	60.49	**77.93**
DistilBERT-Te	63.76	57.60	58.09	75.93
IndicBERT base	64.92	49.29	60.07	75.27
mBERT	65.16	50.01	49.70	75.35
XLM-R	65.76	50.02	49.70	74.71

Table 7.20: p-values for *post hoc* pairwise tests across four NLP tasks. Here, $*$ denotes that the two pairs of models are significantly different (i.e. p 0.05).

Models compared	Sentiment	Emotion	Hate-Speech	Sarcasm
BERT-Te vs *mBERT*	0.0404*	0.032*	0.036*	0.0491*
BERT-Te vs *XLM-R*	0.042*	0.032*	0.036*	0.0482*
BERT-Te vs *IndicBERT*	0.049*	0.046*	0.041*	0.087
IndicBERT vs *mBERT*	0.076	0.121	0.044*	0.092
IndicBERT vs *XLM-R*	0.082	0.193	0.046*	0.112
BERT-Te vs *RoBERTa-Te*	0.188	0.203	0.048*	0.071
BERT-Te vs *DistilBERT-Te*	0.049*	0.253	0.038*	0.087

7.7.3 Do Telugu Pretrained Transformer Models Outperform the State-of-the-art Telugu Datasets?

Here, we fine-tune our Telugu pretrained Transformers and existing multilingual Transformer models on the three state-of-the-art Telugu datasets. The fine-tuning is done independently for each task and pretrained model. We describe the three Telugu datasets as follows:

ACTSA (Sentiment Analysis): We experimented with the publicly available dataset ACTSA [86] to perform the sentiment analysis task. The ACTSA dataset contains the following labels: "Positive", "Negative", and "Neutral".

Article Genre Classification: The task is to predict the genre/topic of a given news article or news headline. We use the news article headline dataset created in IndicCorp [57]. The categories are determined from URL components. The dataset consists of three generic categories which are (i) "entertainment", (ii) "sports", and (iii) "business".

Named Entity Recognition(WikiAnn): We use the Telugu WikiAnn NER dataset [4] created from Wikipedia by utilizing cross language links to propagate English named entity labels to other languages. The WikiAnn NER dataset contains the following coarse grained labels in this dataset: Person (PER), Organisation (ORG), and Location (LOC).

[4]https://elisa-ie.github.io/wikiann/

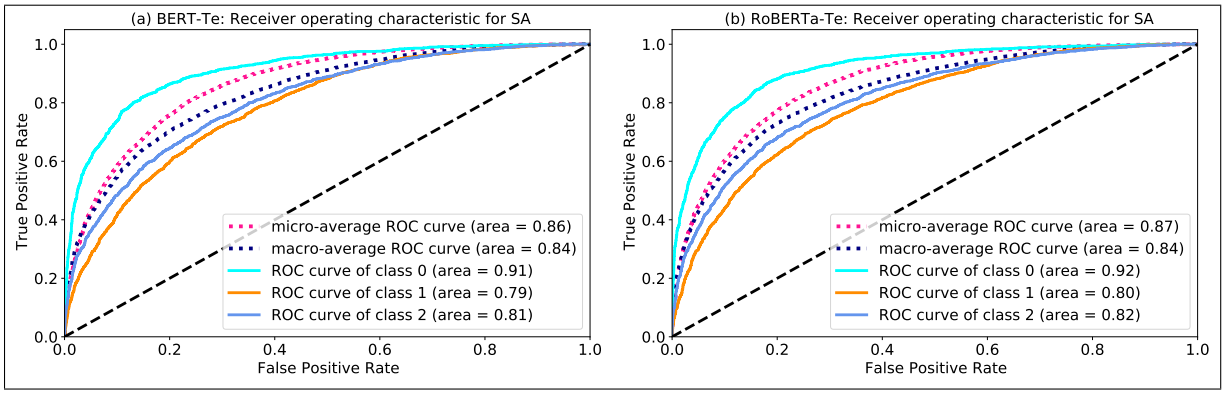


Figure 7.12: ROC curves for SA task: (a) *BERT-Te* and (b) *RoBERTa-Te*. The labels for the three classes are as follows: class 0 - Negative, class 1 - Neutral, and class 2 - Positive.

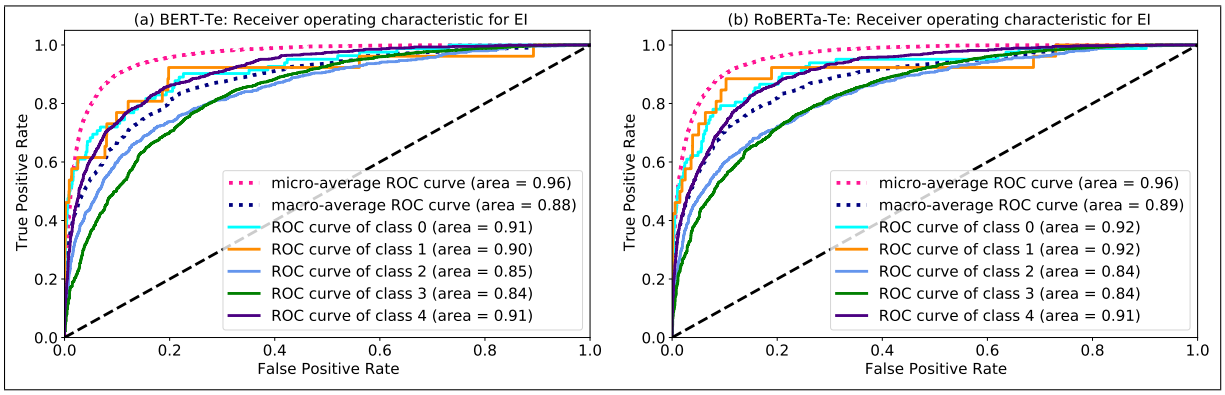


Figure 7.13: ROC curves for EI task: (a) *BERT-Te* and (b) *RoBERTa-Te*. The labels for the five classes are as follows: class 0 - Anger, class 1 - Fear, class 2 - Happy, class 3 - No, and class 4 - Sad.

7.7.4 Performance on the Three Datasets

To further evaluate our Telugu pretrained Transformers, we compare our results in Table 7.21 with the state-of-the-art system used in [57]. Our Telugu pretrained transformers outperform the multilingual models (*mBERT*, *XLM-R*, and *IndicBERT*) on two datasets: Article Genre Classification and WikiAnn. Since *mBERT* model is pretrained on Telugu Wikipedia data, we noticed that the performance of *mBERT* is relatively higher for the tasks based on Wikipedia data, namely WikiAnn NER and Article Genre Classification. Similarly, *IndicBERT* is benefiting from exposure to translated data of multiple Indian languages during pretraining. Hence, *IndicBERT* showcase better performance on ACTSA and Article Genre Classification datasets. However, using the corpora of both Telugu Wikipedia and news websites data, our Telugu pretrained models achieve an accuracy of 99.38% and 87.16% higher compared to the state-of-the-art systems [57]. On the other hand, the pretrained Telugu language models report lower

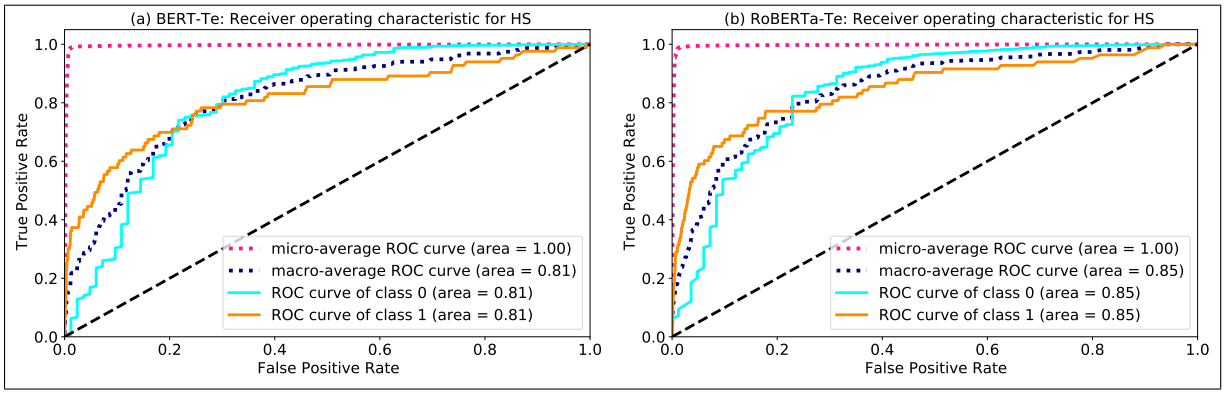


Figure 7.14: ROC curves for HS task: (a) *BERT-Te* and (b) *RoBERTa-Te*. The labels for the two classes are as follows: class 0 - No, and class 1 - Yes.

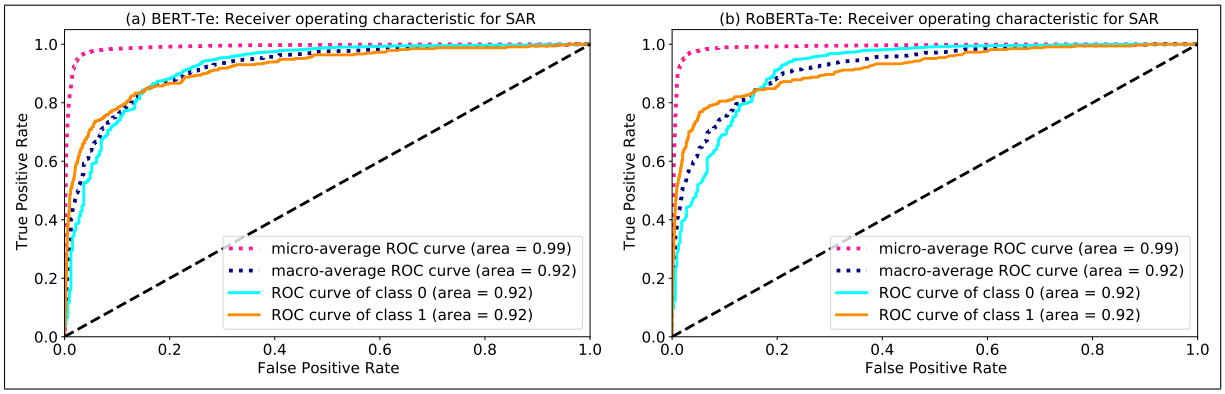


Figure 7.15: ROC curves for SAR task: (a) *BERT-Te* and (b) *RoBERTa-Te*. The labels for the two classes are as follows: class 0 - No, and class 1 - Yes.

performance on ACTSA as compared to *XLM-R* and *IndicERT*. This reinforces the expectation that F1-score is sensitive to large corpora size for the ACTSA task.

7.8 Ethical Statement

We created a Telugu corpus of 80,15,588 sentences. We created different datasets for NLP tasks in the Telugu language. The Telugu NLP datasets are now available for SA, EI, HS, SAR, Clickbait, and NER. We open-source our corpus, five different annotated datasets (SA, EI, HS, SAR, and Clickbait), lexicons, embeddings, and code [5],[6],[7]. The pre-trained Transformer models for Telugu are also made available[8].

[5]https://github.com/Cha14ran/DREAM-T
[6]https://github.com/subbareddy248/Clickbait-Resources
[7]https://github.com/scsmuhio/MTGCN_Resources
[8]https://huggingface.co/ltrctelugu

Table 7.21: Fine-tuning Results on Telugu datasets: Comparison of Text classification results with fine-tuning on existing multilingual pretrained transformers as well as our Telugu pre-trained transformers on three datasets.

	ACTSA	NewsHeadlines	WikiAnn
Feature set↓	F1-score	F1-score	F1-score
BERT-Te	59.40	**99.38**	87.03
ALBERT-Te	46.21	99.22	83.00
RoBERTa-Te	56.75	99.10	**87.16**
DistilBERT-Te	57.86	99.21	84.02
IndicBERT base [57]	61.18	99.20	84.38
mBERT [57]	48.53	98.67	84.31
XLM-R [57]	**62.45**	99.33	81.71

We reused publicly available datasets like ACTSA, IndicCorp, WikiAnn, and LREC-NER to compare state-of-the-art methods. ACTSA [86] is used to perform the sentiment analysis task. The ACTSA dataset contains the following labels: "positive", "negative", and "neutral". The task for Article Genre Classification is to predict the genre/topic of a given news article or news headline. We use the news article headline dataset created in IndicCorp [57]. The categories are determined from URL components. The dataset consists of three generic categories which are (i) "entertainment", (ii) "sports", and (iii) "business". We use the Telugu WikiAnn NER dataset [9] created from Wikipedia by utilizing cross-language links to propagate English named entity labels to other languages (under ODC-By license [10]). The WikiAnn NER dataset contains the following coarse-grained labels in this dataset: Person (PER), Organisation (ORG), and Location (LOC). WikiAnn dataset can be downloaded from [11]. The LREC-NER dataset [12] is licensed under Creative Commons License. Please read their terms of use [13] for more details.

7.9 Privacy Concerns of Crawled Data

We have gone through the privacy policy of various websites mentioned in the thesis. For example, the website privacy policy of greatandhra[14] is provided here[15]. We do not foresee any harmful uses of using the data from these websites.

7.10 Social Impact

Multilingual pre-trained models are usually evaluated by their capacity for knowledge transfer across languages. This can be done either by training the NLP model on English data

[9] https://elisa-ie.github.io/wikiann/
[10] `https://opendatacommons.org/licenses/by/`
[11] `https://drive.google.com/drive/folders/1Q-xdT99SeaCghihGa7nRkcXGwRGUIsKN?usp=sharing`
[12] `http://ltrc.iiit.ac.in/ner-ssea-08/index.cgi?topic=5`
[13] `http://ltrc.iiit.ac.in/ner-ssea-08/index.cgi?topic=5`
[14] `www.greatandhra.com`
[15] `https://www.greatandhra.com/privacy.php`

only or English plus Telugu NLP data using, for example mBERT representations. It allows the model to benefit from high-resource languages. During the testing phase, the NLP model is evaluated in Telugu only. However, this thesis evaluated the NLP model which was only training and testing on Telugu data. The reason for this is to let the models understand the Telugu language.

This thesis creates a large raw Telugu corpus, two large, strongly annotated datasets corresponding to four text classification tasks, and one large annotated dataset for the clickbait task. Further, we propose a supervised graph reconstruction method, Multi-Task Text GCN (MT-Text GCN) on the Telugu, that leverages to simultaneously (i) learn the low-dimensional word and sentence graph embeddings from word-sentence graph reconstruction using graph autoencoder (GAE) and (ii) perform multi-task text classification using these latent sentence graph embeddings. Our proposed MT-Text GCN and feature representations achieve significant improvements on developed datasets over existing multilingual pre-trained Transformer models: mBERT [34], and XLM-R [28].

Our investigation neither introduces any social/ethical bias to the model nor amplifies any bias in the data. We do not foresee any direct social consequences or ethical issues.

7.11 Summary

The resource-poor Indian languages (e.g., Telugu) is not much matured in comparison to resource-rich languages such as English due to a lack of sufficient resources. In this chapter, we presented our contributions for Telugu: (i) creation of a large corpus to generate contextual Telugu word embeddings/sentence embeddings from scratch, (ii) creation of large annotated data for four NLP tasks, (iii) creation of lexicons for sentiment, emotion, and hate-speech. (iv) creation of Telugu pre-trained transformer representations. The large Telugu corpora (8 million sentences), annotated data of four NLP tasks, pre-trained word, and sentence embeddings, pre-trained Telugu transformers are publicly available here[16] and the pre-trained trandformer models are available here [17].

In the next chapter, we present the dataset and models for automatic clickbait detection in Telugu language.

[16]https://github.com/Cha14ran/DREAM-T
[17]https://huggingface.co/ltrctelugu

Chapter 8

Dataset, Embeddings and Models for Clickbait Detection

This chapter explains the process of creating our annotated clickbait dataset, feature representations from scratch, for building an automatic clickbait detection system.

8.1 Background

Clickbait headlines have become a nudge in social media and news websites. The methods to identify clickbaits are mainly being developed for English. There is a need for the same in other languages with the increase in the usage of social media platforms in different languages. In this work, we present an annotated clickbait dataset of 112,657 headlines that can be used for building an automated clickbait detection system for Telugu, a resource-poor language. Our contribution in this chapter includes (i) generation of the latest pre-trained language models, including RoBERTa, ALBERT, and ELECTRA trained on a large Telugu corpus of 8,015,588 sentences that we had collected, (ii) data analysis and benchmarking of the performance of different approaches ranging from hand-crafted features to state-of-the-art models. We show that the pre-trained language models trained in Telugu outperform the existing pre-trained models, viz. BERT-Mulingual-Case [34], XLM-MLM [70], and XLM-R [28] on clickbait task. On a large Telugu clickbait dataset of 112,657 samples, the Light Gradient Boosted Machines (LGBM) model achieves an F1 score of 0.94 for clickbait headlines. For Non-Clickbait headlines, an F1 score of 0.93 is obtained, similar to that of the Clickbait class. We open-source our dataset, pre-trained models, and code[1]. Our contributions to this work can be summarized as follows:

- We publicly released an annotated dataset of 112,657 Telugu clickbait and non-clickbait headlines that can be a crucial resource for building automated clickbait detection systems in Telugu.

[1]https://github.com/subbareddy248/Clickbait-Resources

- We generated the three latest pre-trained language models, including RoBERTa [75], ALBERT [71], and ELECTRA [26] for Telugu language, trained on large Telugu corpus (8,015,588 sentences) from scratch.

- We developed a benchmark system for detecting clickbait headlines written in Telugu by investigating a wide range of features from traditional to state-of-the-art representations.

- We explored the feasibility of different neural architectures and pre-trained transformer models in this problem.

- We presented a detailed analysis of the methods and results and compare the existing pre-trained embeddings for detecting clickbait headlines in Telugu.

8.2 A New Dataset for Detecting Clickbait in Telugu

We followed certain guidelines for collecting the data extensively for clickbait and non-clickbait headlines.

Inter-Annotator Agreement The annotation process on sentences for clickbait is a subjective task. In the guidelines to annotators, we defined the attractiveness of clickbait headlines as something that generates interest, arouses curiosity by using suspenseful language there, and motivates the user to click on the link. Moreover, the attractiveness of the sentences is related to one's interests and curiosity. We made the five annotators work on a smaller dataset for verification as a first step of performing the annotation. The Fleiss' kappa score[2] was 0.95. Then the annotation on the whole dataset was done, for which the inter-annotator agreement was 0.91.

Clickbait vs Non-Clickbait: To collect the clickbait and non-clickbait headlines, we manually selected four Telugu websites, including, Prajasakti[3], Gulte[4], Andhrabhoomi[5], Manatelangana[6] which publish gossips, trends, and latest buzzes. While collecting misleading headlines from these sites, we found that most sites have the same headlines. To avoid the redundant text, we considered the Jaccard similarity for discarding duplicate sentences (above 90% match) and reported 112,657 headlines. To filter the non-clickbait (i.e., the headlines in these domains which are not attractive to users), we provided the data and a web-based framework to an *Elancer IT Solutions Private Limited*[7], an annotation company for labeling the headlines. We choose five native Telugu language speakers from *Elancer IT Solutions Private Limited* company to

[2]https://en.wikipedia.org/wiki/Fleiss%27_kappa
[3]http://www.prajasakti.com
[4]https://telugu.gulte.com
[5]http://www.andhrabhoomi.net
[6]https://www.manatelangana.news/
[7]http://elancerits.com/

annotate headlines as clickbait and non-clickbait. Since five annotators label each headline, we obtained a 'substantial' inter-annotator agreement with a Fleiss' kappa score of 0.91.
Dataset Statistics Taking the majority vote as ground truth, 79,190 headlines were marked as clickbait and 33,467 as non-clickbait.

8.3 Feature Representation Methods

This section discusses the detailed analysis of various feature representation methods ranging from hand-crafted to automated features to develop the models. To perform better training, we use the romanized text format, i.e., whole Telugu corpus and annotated data converted into WX notation[8]. In this chapter, we use Polyglot tokenizer[9] for tokenizing the sentences.

8.3.1 Traditional Features

To extract the traditional features like Bag-Of-Words (BoW) and Term frequency-Inverse document frequency (TF-IDF), we built a feature set that resulted in a dataset size of 112,657 (samples) x 2000 (features).

8.3.2 Hand-Crafted Features

Structural Features Here, we will discuss how the sentence structure and punctuation features can help discriminate between clickbait and non-clickbait headlines. From the Table 8.1, we observe that frequency of the presence of exclamation marks (!), question marks (?), word length < 4, and Hapax Legomena Ratio ($\frac{\text{number of unique words occurred only once}}{\text{total number of words}}$) are high in clickbait headlines. However, the average length of the sentence and the number of stop words display high values for non-clickbait.

Table 8.1: Analysis of structural features

Average occurences	Clickbait	Non-Clickbait
Average length	4.98	12.25
Average stopwords	0.51	2.28
Average Hapax Legomena Ratio	0.13	0.10
Average Dis Legomena Ratio	0.03	0.01
Ratio of sentences with word length<4	0.17	0.16
Ratio of sentences with word length>6	0.02	0.02
Ratio of sentences with Exclamation	0.15	0.01
Ratio of Sentences with Question mark	0.12	0.02

Part-of-Speech (POS) Features The POS tagging features are often useful for understanding particular patterns in morphologically complex languages like Telugu. We use a feature set that

[8]https://github.com/irshadbhat/indic-wx-converter
[9]https://github.com/ltrc/polyglot-tokenizer

Table 8.2: Analysis of POS Tag features

Pos Tag Features	Clickbait	Non-Clickbait
Average nouns per sentence	2.60	5.62
Average verbs per sentence	0.61	2.14
Average adjectives per sentence	0.07	0.25
Average adverbs per sentence	0.08	0.35
Average pronouns per sentence	0.09	0.34
Average postpositions per sentence	0.06	0.22
Average punctuations per sentence	0.94	1.29

includes the number of nouns (NN), number of verb mains (VM), number of symbols (SYM), number of adverbs (RB), number of prepositions (PSP), number of adjectives (JJ), and number of pronouns (PRP) from each sentence using part-of-speech (POS) tagging[10]. Table 8.2(b) reports the average number of times each POS tag occurs in the two kinds of headlines.

8.3.3 Distributed Word Representations

Distributed word representations capture many precise syntactic and semantic word relationships in text classification problems. This subsection describes the generation of different word embeddings such as Word2Vec, GloVe, FastText, and Meta-Embeddings on large Telugu corpora (Wikipedia + Crawled corpus) consisting of 8,015,588 sentences.

Word2Vec Embeddings Word2Vec model provides a non-deterministic way to determine the word representations [80]. It can learn similar word vectors for words in a similar context. We used the Skip-Gram (SG) Word2vec [80] model with a large vocabulary size of 460,000 words, where each word frequency is at least 5. The number of linear neurons in the hidden layer is 300, and the context window size is five as the hyper-parameters used for the model training.

GloVe Embeddings The input used in the GloVe model is a non-zero word-word co-occurrence matrix [95], which adds the global context information by default, unlike the use of local context in Word2Vec [80]. We use the parameters with a learning rate of 0.05, the context window size is 10, and the word vector dimension is 50.

FastText Embeddings Since the FastText model considers the bag of character n-grams to represent each word [56], it allows us to compute rare word representations. We considered the words with a frequency of at least three in the vocabulary and obtained a vocabulary size of 462,232 in the training process. We use a successful Skip-Gram model with a context window size of 5, and the word vector dimension is 200.

Meta-Embeddings Meta-embeddings are shown to be successful for resource-rich language English, and it has two benefits compared to individual embedding sets: (i) enhancement of performance and (ii) improved coverage of the vocabulary [137]. These prior studies motivate us to create Meta-Embeddings for the Telugu language by using an ensemble of Telugu Word2Vec, GloVe, and FastText embeddings. In this work, we created the Meta-Embeddings using the

[10]https://bitbucket.org/sivareddyg/telugu-part-of-speech-tagger/src

average of encoded source embeddings that yields each word vector dimension of 300 used for extracting the text features.

8.3.4 Skip-Thought Sentence Vectors

The Skip-Thought model is a sentence encoding-decoding model that produces a fixed-length vector for every given sentence [67]. Inspired by the Word2Vec approach, the skip-thought model extends the Skip-Gram model to generate sentence embeddings from word embeddings. Rather than predicting the context given the input word, the skip-thought model predicts both the next and previous sentences given the target sentence.

8.3.5 Context Level Features

Earlier word embeddings such as Word2Vec, GloVe, FastText, and Meta-Embeddings provide a unique word representation throughout the corpus. This unique representation is the main limitation of these methods, especially words with different contexts. To overcome these limitations, in this section, we describe the recent state-of-the-art pre-trained models, including ELMo, BERT, RoBERTa, ALBERT, and ELECTRA. The parameters used for pre-training the models on Telugu are reported here[11].

ELMo Embeddings for Telugu Embeddings from Language Models (ELMo) is a successful NLP framework developed by AllenNLP [97] group. Unlike earlier embeddings, the ELMo embeddings represent the words contextually using a bidirectional LSTM model. We generate ELMo embeddings for the Telugu language, trained on a large Telugu corpus with 2,086,488 train tokens and a vocabulary size of 793,384.

BERT Embeddings for Telugu BERT model [33] provides a word contextual information by looking at previous and next words, which is one of the main limitations of earlier methods. To train the BERT model from scratch for the Telugu language corpus, we build a vocabulary size of 832,000 using the BERTWordPiece tokenizer. We use the hidden representation associated with the CLS token as a numeric representation of the given sentence.

RoBERTa Embeddings for Telugu The recent successful RoBERTa model [75] is an optimized pre-training language model that improves on BERT, the self-supervised method that achieves state-of-the-art results. The main objectives of RoBERTa model include (i) training the model longer, with bigger batches, over more data; (2) removing the following sentence prediction approach; (3) training on longer sequences; and (4) dynamically changing the masking pattern applied to the training data. Here, we use a vocabulary size of 200,000 and a hidden word dimension of 768 when training RoBERTa masked language model on the Telugu corpus.

ALBERT Embeddings for Telugu ALBERT (A lite BERT) [71] model is a novel pre-training method that mainly contributed to three solutions to overcome limitations of the BERT model,

[11] https://github.com/subbareddy248/Clickbait-Resources

including (i) parameter reduction using factorized embedding parameterization, (ii) Cross-layer parameter sharing, and (iii) Inter-sentence coherence loss. With the advantage of a minimal number of parameters and yet achieve state-of-the-art results than the BERT model [71]. To train ALBERT embeddings for Telugu, we use a subword-byte level tokenizer to build the vocabulary size of 105,686 with the hidden embedding dimension is of 768.

Table 8.3: Clickbait Detection Results: Different feature sets classification comparison for Logistic Regression, and LightGBM models using different sampling methods with No/Over/Under-Sampling

	Logistic Regression									LightGBM								
Sampling Method→	NS			OS			US			NS			OS			US		
Feature set↓	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
BoW	0.87	0.82	0.84	0.83	0.84	0.84	0.84	0.84	0.83	0.89	0.80	0.83	0.84	0.84	0.84	0.83	0.84	0.83
TF-IDF	0.86	0.82	0.83	0.81	0.83	0.82	0.81	0.83	0.82	0.90	0.81	0.84	0.82	0.83	0.83	0.81	0.83	0.82
Structure-Based	0.90	0.87	0.88	0.87	0.88	0.87	0.87	0.88	0.87	0.90	0.87	0.88	0.87	0.88	0.87	0.87	0.88	0.87
Pos Tagging Set	0.90	0.87	0.88	0.87	0.88	0.87	0.87	0.88	0.87	0.90	0.87	0.88	0.87	0.88	0.87	0.87	0.88	0.87
Word2Vec	0.75	0.73	0.74	0.74	0.77	0.75	0.74	0.77	0.75	0.90	0.88	0.89	0.89	0.89	0.89	0.86	0.88	0.87
GloVe	0.66	0.63	0.63	0.67	0.69	0.67	0.67	0.70	0.67	0.83	0.81	0.82	0.82	0.83	0.82	0.79	0.82	0.80
FastText	0.78	0.77	0.77	0.76	0.80	0.77	0.76	0.80	0.77	0.90	0.89	0.89	0.90	0.89	0.90	0.87	0.89	0.88
Meta-Embeddings	0.75	0.73	0.74	0.74	0.78	0.75	0.73	0.76	0.74	0.91	0.89	0.90	0.90	0.90	0.90	0.88	0.90	0.88
Skip-Thought	0.92	0.90	0.91	0.91	0.91	0.91	0.89	0.90	0.90	0.92	0.90	0.91	0.91	0.91	0.91	0.89	0.91	0.90
BERT	0.89	0.86	0.87	0.86	0.87	0.86	0.85	0.88	0.86	0.90	0.87	0.89	0.88	0.88	0.88	0.86	0.88	0.87
ALBERT	0.89	0.84	0.86	0.85	0.85	0.85	0.85	0.85	0.85	0.89	0.85	0.86	0.86	0.86	0.86	0.86	0.86	0.86
RoBERTa	0.93	0.91	0.92	0.91	0.92	0.91	0.90	0.91	0.90	0.93	0.91	0.92	0.92	0.91	0.92	0.90	0.92	0.91
ELECTRA	0.88	0.86	0.87	0.87	0.87	0.87	0.86	0.87	0.86	0.91	0.89	0.90	0.90	0.90	0.90	0.88	0.90	0.89
ELMo	0.95	0.94	0.94	**0.94**	**0.94**	**0.94**	0.93	0.94	0.93	0.95	0.93	0.94	**0.94**	**0.94**	**0.94**	0.93	0.94	0.94

NS = No-Sampling, OS = Over-Sampling, US = Under-Sampling, P = Precision, R = Recall, F1 = F1-score

ELECTRA Embeddings for Telugu Recently developed language models such as BERT [33], ALBERT [71], and RoBERTa [75] fall under the category of masked language models that predict words that have been masked out of the input. Thus, masked language models only predict a small subset — the 15% that was masked out, reducing the amount learned from each sentence.

The recent state-of-the-art model ELECTRA uses a new pre-training approach, called replaced token detection (RTD), that trains a bidirectional model (like a Masked Language Model(MLM)) while learning from all input positions (like an LM). The ELECTRA model used the GANs approach to distinguish between "real" and "fake" input tokens in the training process. It corrupts the input by replacing some input tokens with incorrect but plausible fakes. We create ELECTRA-small embeddings for the Telugu language, trained on a Telugu corpus with a vocabulary size of 388,292 and a hidden dimension of 256.

8.4 Methodologies

We experimented with the two off-the-shelf machine learning classifiers, including, logistic regression (baseline) [138], and LightGBM [59] to evaluate our clickbait dataset. We also experimented with a deep learning architecture LSTM [49] to perform clickbait classification.

Logistic Regression (LR) We train a Logistic Regression classifier on the training data with regularization by passing the parameters: {C = 20.0, dual = False, penalty = l2}. Formally, the input to the logistic regression model [138] is a text vector extracted from one of the feature representation methods shown in Section 8.3.

LightGBM We choose the LightGBM model [59] as one of our training methods because of its high speed and consumption of less memory on large datasets. To train the LightGBM model, we used the same input feature representations as the LR model. We pass the hyper-parameters: {max_depth = 1, min_child_samples = 13, number of decision trees = 100, boosting type = gradient boosting, learning_rate = 1.0} when training the model.

Long Short-Term Memory Networks (LSTM) In the literature, LSTMs [49] are successful in dealing with sequence-based problems in the field of NLP [148]. Using LSTM architecture, we tried the five different feature representations as input in the training process. We train each LSTM model separately for extracted token features and four-word embeddings.

Choice of Hyper-Parameters We used genetic algorithm for hyper-parameter tuning [72] to optimize both LR and LightGBM models. The hyper-parameters which resulted in the best F1 score were considered.

Imbalanced Dataset Handling From section 8.2, we observe that clickbait data has a class imbalance issue, as the majority class ("Yes") has ∼3x more instances compared to the "No" class. To circumvent this problem, we tried methods such as random undersampling and SMOTE (Synthetic Minority Oversampling Technique) [19] to generate synthetic samples for the minority class (non-clickbait dataset) in the over-sampling method.

8.5 Experiments and Results

Dataset Splitting We followed the stratified 5-fold cross-validation setup where 4-folds data is used for training, and one-fold is used for testing the model. We calculated the average of 5-folds and reported the results.

Evaluation Metrics We use classification metrics such as macro-average precision, recall, and F1-score to evaluate our methods. To understand how each class performs, we choose macro averaging, which gives each class equal weight to evaluate systems performance across the two classes.

8.5.1 Results and Analysis

The results for our 14-feature representations with two machine learning models are reported in Table 8.3. Impressively, the ELMo-Telugu features outperform all the feature representations, with the best model, LightGBM, in the over-sampling setting yielding an F1 score of 0.94. We observed the following observations from Table 8.3.

Baseline Results: We considered the two traditional feature representations (BoW and TF-IDF) as our baseline features. Since the traditional features do not capture the semantic and syntactic information, the baseline system achieves an F1 measure of 0.84.

Hand-Crafted Feature Results: Hand-crafted features such as Structure-based and POS-tagging outperforms the baseline features with an increasing F1 score of 0.04.

Word-Vector Results: The four rows in the third block of the Table 8.3 employ the word-embedding results when passing the input features as Word2Vec, GloVe, FastText, and Meta-Embeddings. With the local and global context word-embeddings, the system achieves an F1-measure of 0.89 (Word2Vec) and 0.82 (GloVe), using the LightGBM method over-sampling strategy. Like Word2Vec, GloVe, FastText and Meta-Embedding features as input yield an F1-score of 0.90 using the LightGBM method in the over-sampling strategy. Among the four word-embeddings, FastText, and Meta-Embedding features improve the F1 measure by 0.06 compared with baseline. However, the performance of word embedding results when trained with Logistic Regression is lower than other methods. To interpret the results of LR and LightGBM models for Word2Vec, we use the Local Interpretable Model-Agnostic Explanations (LIME) [108] tool to showcase the highlighted words causing the model toward clickbait or non-clickbait prediction in the sub-section 8.5.4.

Skip-Thought and Contextual Embeddings Results: Using the sentence embedding method Skip-Thought performed the best with an F1 score of 0.91, higher than the methods mentioned above. Overall, LGBM provides better results across BERT-based models with similar performance. The best F1-score is obtained when RoBERTa (0.92) and ELECTRA (0.90) feature sets are used. ELMo increases the system's performance to a 0.94 F1 score, higher than all feature methods with contextual embedding features.

Table 8.4: Clickbait Detection Results: The table display the fine tuning results of the existing pre-trained language models

	Fine-Tuning		
Models↓	Precision	Recall	F1-score
BERT	0.89	0.86	0.87
ALBERT	0.89	0.84	0.86
RoBERTa	0.91	0.87	0.89
ELECTRA	0.88	0.86	0.87
Bert-Multilingual-case	0.74	0.79	0.74
XLM-Roberta-base	0.78	0.83	0.79

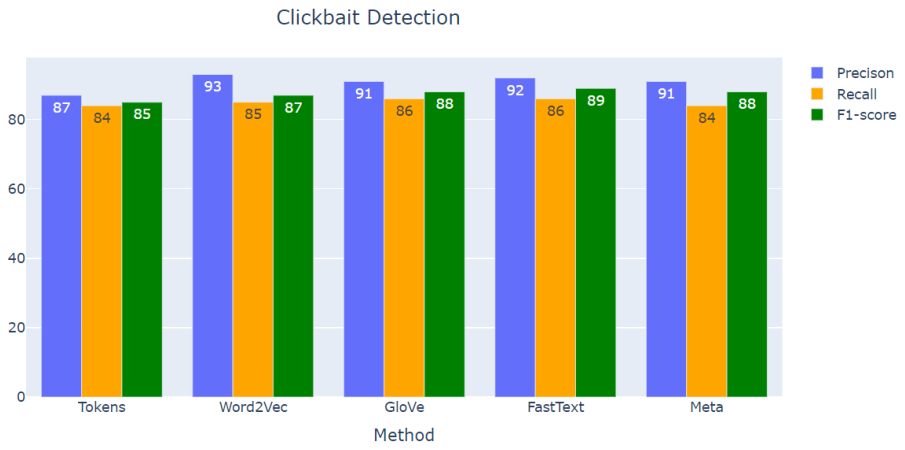


Figure 8.1: Comparison of F1-score performance of input word representations i) Sequence of Tokens ii) Word2Vec iii) GloVe iv) FastText and v) Meta-Embedding. The LSTM model is used as a trained model for each of the feature representations in the 2-class setups are shown here

LSTM Results: Figure 8.1 showcase the clickbait detection results of LSTM trained on (i) sequence of tokens, and (ii) with the four-word embeddings. From Figure 8.1, we observe that LSTM with the FastText features as input yields a better F1-score of 0.89 similar to the Table 8.3. Moreover, GloVe word vectors as input to LSTM report a 0.88 F1 score higher than the F1 score displayed in Table 8.3.

Overall, except for distributed word-embedding results, the remaining pre-trained models perform similarly to Logistic Regression and LightGBM models.

Fine-Tuning Results: We evaluate whether fine-tuning the existing pre-trained language models (BERT-Multilingual-case, XLM-Roberta-base) and the pre-trained models built on Telugu is useful for adapting the click-bait task. From Table 8.4, we find that fine-tuning results show better performance than earlier word embeddings and baselines. However, extracting the features from pre-trained language models has increased performance as it may allow us to adapt general-purpose representations, as reported in Table 8.3.

8.5.2 Comparative Analysis

Figure 8.2 showcase the comparative results on clickbait-detection task, where the features extracted from existing pre-trained embeddings trained on the Telugu language, such as Bert-Multilingual-Case [34] (768-dimension), Cross-Lingual Language Model with RoBERTa (XLM-R) [28] (1024-dimension), Cross-Lingual Masked Language Model (XLM-MLM) [70] (1280-dimension), Sub-word Byte Pair embeddings [48] (300-dimension), and Facebook FastText word-embeddings [13] (200-dimension). We compare the embeddings BERT, RoBERTa, ALBERT, ELECTRA, and ELMo trained on Telugu corpus (Wikipedia + Crawled Data) with

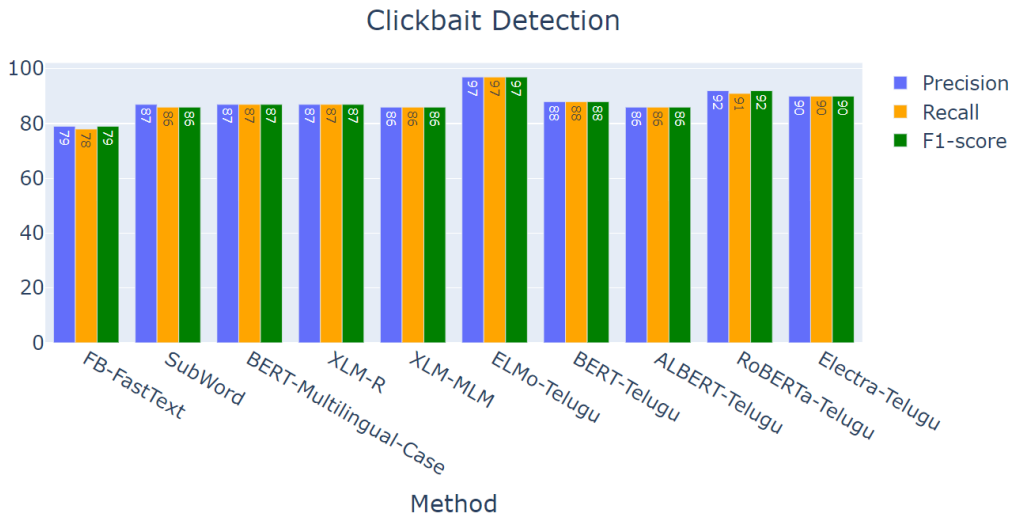


Figure 8.2: Comparison of F1-score performance of i) Facebook FastText, ii) Subword Byte Pair iii) BERT-Multilingual-Case iv) XLM-R v) XLM-MLM vi) ELMo-Telugu vii) BERT-Telugu viii) ALBERT-Telugu ix) RoBERTa-Telugu and x) ELECTRA-Telugu embeddings. The LightGBM classifier is used to train each of the feature representations in the clickbait task setup is shown here

the above-mentioned pre-trained embeddings. Our generated pre-trained embeddings for the Telugu language outperform the existing pre-trained embeddings in the two-class clickbait-detection task. Our best model's performance is from the ELMo feature representation with an F1-score of 0.94, an improvement of 0.05 than existing pre-trained methods. These results indicate that the performance of the generated pre-trained embeddings BERT, ELMo, RoBERTa, and ELECTRA on Telugu corpus would have been much better than existing pre-trained embeddings.

Table 8.5: ELMo: Confusion matrix for clickbait classification

		Predicted	
		Non-Clickbait	Clickbait
Actual	Non-Clickbait	6092	545
	Clickbait	604	15291

8.5.3 Error Analysis

We analyzed the error cases in detail for the clickbait and non-clickbait samples. We observed from Table 8.5 that (i) 3.8% cases clickbait headlines predicting as non-clickbait, and (ii) 8.2% cases non-clickbait headlines predicting as clickbait. Our system makes incorrect predictions where the sentences contain the implicatures. Table 8.6 display the failure cases where the model got confused due to the pragmatic notations like "Implicatures", "Deictics", and "Noun Ellipsis". We report the detailed confusion matrices for contextualized word embeddings

Table 8.6: ELMo: Wrong predictions by the model

Headline	Act	Pred	Reason
వృద్ధాప్య లక్షణాలను దూరం చేయడానికి ఇవి ఫాలో అవ్వండి.... vqxXApya lakRaNAlanu xUraM ceyadAniki.... Follow these to get rid of the signs of aging ...	yes	no	Model could not understand the implicature
వాటిలో తెలంగాణ పోలీసులు ముందున్నారు... vAtilo weVlaMgANa polIsulu muMxunnAru... Among them, the Telangana police are leading ...	yes	no	Model could not understand the deictic information
భిన్న కోణాల్లో దర్యాప్తు చేస్తున్న పోలీసు బృందం... Binna koNAllo xaryApwu ceswunna polIsu bqMxa... Police team investigating from different angles ...	no	yes	Model guessed Noun ellipsis as clickbait

Table 8.7: BERT: Confusion matrix for clickbait classification

		Predicted	
		Non-Clickbait	Clickbait
Actual	Non-Clickbait	5510	1184
	Clickbait	1027	14811

(BERT, ALBERT, RoBERTa, and ELECTRA) in Tables 8.7, 8.8, 8.9,and 8.10 respectively. Observations from the Tables 8.7, 8.8, 8.9,and 8.10 that both ELECTRA and RoBERTa models have low false positives compared to BERT and ALBERT.

8.5.4 LIME Results

Figure 8.3 showcases the words highlighted for both LR and LightGBM models using LIME. We can observe from Figure 8.3 that the LightGBM model captures attractive words when compared with the LR model. Although the input sentence in Table 8.11 is a clickbait, the LR model predicted it as non-clickbait, whereas the LightGBM model predicted it correctly.

8.5.5 Statistical Analysis

Statistical significance using one-Anova test [60] showcases the significant difference between the 14 feature representations used in the above experiments. We perform the statistical significance test on a clickbait-detection task with a two-class setup. To perform the Anova test, we use the three-sampling results (F1-measure) of two training classifiers, LR and LightGBM. The one-way Anova test provides an F-statistic [$F(13,70) = 9.8035$, $P = 2.9e^{-11}$] concludes 14 feature representations are significantly different. We use the post-hoc Tukey-HSD test [109] to obtain the results between different pairs reported in Table 8.12.

Table 8.8: ALBERT: Confusion matrix for clickbait classification

		Predicted	
		Non-Clickbait	Clickbait
Actual	Non-Clickbait	5367	1327
	Clickbait	1375	14463

Table 8.9: RoBERTa: Confusion matrix for clickbait classification

		Predicted	
		Non-Clickbait	Clickbait
Actual	Non-Clickbait	5776	918
	Clickbait	604	15234

Table 8.10: ELECTRA: Confusion matrix for clickbait classification

		Predicted	
		Non-Clickbait	Clickbait
Actual	Non-Clickbait	5412	1282
	Clickbait	565	15273

8.6 Summary

In this chapter, we present a new annotated dataset of approximately 113k headlines that can be used for building an automated clickbait detection system for a resource-poor language Telugu. The evaluation of two machine learning classifiers and LSTM-based models suggest that simple machine learning classifiers with ELMo features perform better than the baseline and LSTM-based models. As a future task, we aim to look at the strength of clickbait (going beyond the binary class) and explore related tasks like fake news identification or sarcasm detection.

In the next chapter, we present the summary, conclusions, limitations and future work of the thesis.

Table 8.11: Lime: Input sentence in Telugu, WX, and English

Headline
ఎూర్య 24.. ఏంటి కథ?.
sUrya 24.. eMti kaWa?
Surya 24 .. What story ?.

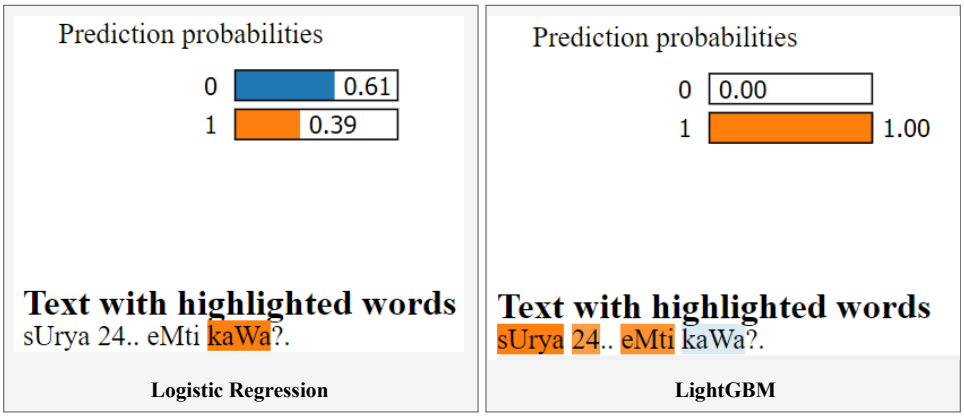


Figure 8.3: Local Interpretable Model-Agnostic explanations: showcase the words highlighted for both LR and LightGBM models

Table 8.12: Post-hoc Tukey test results

Feature Pairs	P-value	Statistically Significant
(ALBERT vs. GloVe)	0.001	Yes
(BERT vs. GloVe)	0.001	Yes
(BOW vs. ELMo)	0.001	Yes
(BOW vs. GloVe)	0.0081	Yes
(ELECTRA vs. GloVe)	0.001	Yes
(ELMo vs. FastText)	0.001	Yes
(ELMo vs. Meta-Embeddings)	0.001	Yes
(ELMo vs. TF-IDF)	0.001	Yes
(GloVe vs. POS Tags)	0.001	Yes
(GloVe vs. Structure Based)	0.001	Yes
(ALBERT vs. Meta-Embeddings)	0.9	No
(Structure Based vs. Pos Tags)	0.9	No
(BoW vs. ELECTRA)	0.792	No
(BoW vs. TF-IDF)	0.9	No
(TF-IDF vs. Word2vec)	0.9	No
(RoBERTa vs. Word2vec)	0.0101	No
(ELECTRA vs. TF-IDF)	0.5856	No
(FastText vs. Skip-Thought)	0.1215	No
(Skip-Thought vs. Word2vec)	0.0232	No

Chapter 9

Summary, Conclusions and Future Work

In this chapter, the summary, conclusions and future work of the thesis is presented.

9.1 Summary

In this thesis, our primary goal is to solve the data scarcity problem for Telugu, a low-resource problem. We present several contributions to overcome the problem: (1) A large Telugu raw corpus of 80,15,588 sentences (16,37,408 sentences from Telugu Wikipedia and 63,78,180 sentences crawled from different Telugu websites). (2) A TEL-NLP annotated dataset in Telugu covering four NLP tasks (16,234 samples for sentiment analysis (SA), hate-speech detection (HS), sarcasm detection (SAR), and 9,675 samples for emotion identification (EI)). (3) For the Telugu corpus, we are the first to generate word and sentence embeddings using graph-based models: *DeepWalk-Te* and *Node2Vec-Te*, and *Graph AutoEncoders (GAE)*. (4) Also, we propose the multi-task learning model (MT-Text GCN) to reconstruct word-sentence graphs on TEL-NLP data while achieving multi-task text classification with learned graph embeddings. (5) We extended samples in the annotated dataset (35,142 sentences in each task) for multiple NLP text classification tasks such as SA, EI, HS, and SAR. (6) We create different lexicons for sentiment, emotion, and hate-speech for improving the efficiency of the machine learning models. (7) From scratch, we pre-trained distributed word and sentence embeddings such as *Word2Vec-Te*, *GloVe-Te*, *FastText-Te*, *MetaEmbeddings-Te*, *Skip-Thought-Te*. (8) We pre-trained different contextual language models for Telugu such as *ELMo-Te*, *BERT-Te*, *RoBERTa-Te*, *ALBERT-Te*, *Electra-Te*, and *DistilBERT-Te* trained on 80,15,588 Telugu sentences. (9) We created a clickbait dataset of 112,657 samples to create an automatic clickbait detection in Telugu.

Further, we show that these representations significantly improve the performance of five NLP tasks and present the benchmark results for Telugu. We argue that our pre-trained embeddings are competitive or better than the existing multilingual pre-trained models: *mBERT*, *XLM-R*, and *IndicBERT*. Lastly, the fine-tuning of pre-trained models show higher performance than linear probing results on five NLP tasks. We also experiment with our pre-trained

models on other NLP tasks available in Telugu viz. Named Entity Recognition, Article Genre Classification, and, Sentiment Analysis and find that our Telugu pre-trained language models (*BERT-Te* and *RoBERTa-Te*) outperform the state-of-the-art system except for the sentiment task. We hope that the availability of the created resources for different NLP tasks will accelerate Telugu NLP research which has the potential to impact more than 85 million people. It can also help the Telugu NLP community evaluate advances over more diverse tasks and applications. We open-source our corpus, five different annotated datasets (SA, EI, HS, SAR, and Clickbait), lexicons, embeddings, pre-trained transformer models, and code.

9.2 Conclusions

The resource-poor Indian languages (e.g., Telugu) are not much mature compared to resource-rich languages such as English due to insufficient resources. In our thesis, we made several contributions to Telugu (i) creation of a large corpus to generate contextual Telugu word embeddings, sentence, and graph embeddings from scratch, (ii) creation of large annotated data for five NLP tasks, (iii) creation of lexicons for sentiment, emotion, and hate-speech. (iv) creation of Telugu pre-trained Transformers. (v) creation of Multi-task GCN architecture to improve the performance in Telugu. Our results on five NLP tasks improve over different existing pre-trained embeddings. We open-source our corpus, five different annotated datasets (SA, EI, HS, SAR, and Clickbait), lexicons, embeddings, and code [1]. The pre-trained Transformer models for Telugu are available here[2].

9.3 Limitations

The main idea of this thesis is to create resources for Telugu. Also, build a corpus, annotated datasets, feature representations, and models for the Telugu language. We have proposed a few architectures for these annotated datasets. The results are encouraging when compared with the existing multi-lingual representations and models. However, we observed a few limitations to the thesis and mentioned below:

- To start with the Telugu language, we created resources and then annotated datasets for the basic NLP tasks. However, there is a scope to go deep into each task and develop datasets for fine grained tasks.
- We considered experiments with few tasks as they are the popular NLP tasks in English. In addition, to these tasks, other NLP tasks like summarization, inference tasks, and question answering can be considered.

[1] https://github.com/mounikamarreddy/NLP-for-Telugu-Language.git
[2] https://huggingface.co/ltrctelugu

- There is a scope to do the experiments with multiple Indian languages and then compare the results. We cannot compare as no existing datasets are available for these tasks in other languages. There is scope to compare the experimental results in a multi-lingual setting.

9.4 Future Work

The proposed work has many directions for future work, a few of them are as follows:

- We want to include datasets for more fine-grained tasks like aspect-based sentiment analysis, target aspect-based sentiment analysis, fine-grained emotion identification, fake news detection, bizarre news detection, and so on.
- One aspect is how easily we can map any natural language tasks into a human-readable prompted form, which should lead to zero-shot generalization.
- It would be interesting to perform cross-task generalization on unseen datasets by adding more task instructions to each dataset.
- We would like to evaluate how the knowledge transfer from Telugu helps the other Dravidian languages.
- Experiments can reveal the understanding of what each layer of the pre-trained language model learns related to the structure of the Telugu language.
- Experiments to conduct the interpretability of pre-trained language models where the language model is different w.r.t to each fine-tuned task.

Bibliography

[1] M. Abdul-Mageed and L. Ungar. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 718–728, 2017.

[2] A. Agrawal. Clickbait detection using deep learning. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, pages 268–272. IEEE, 2016.

[3] S. Amir, B. C. Wallace, H. Lyu, and P. C. M. J. Silva. Modelling context with user embeddings for sarcasm detection in social media. *arXiv preprint arXiv:1607.00976*, 2016.

[4] R. T. Anchiêta, F. A. R. Neto, R. F. de Sousa, and R. S. Moura. Using stylometric features for sentiment classification. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 189–200. Springer, 2015.

[5] S. Arulmozi and M. K. Murty. Building telugu wordnet using expansion approach. In *The WordNet in Indian Languages*, pages 201–208. Springer, 2017.

[6] S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining.

[7] Y. Bao, Q. Ma, L. Wei, W. Zhou, and S. Hu. Multi-granularity semantic aware graph model for reducing position bias in emotion-cause pair extraction. *arXiv preprint arXiv:2205.02132*, 2022.

[8] F. Barbieri, H. Saggion, and F. Ronzano. Modelling sarcasm in twitter, a novel approach. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 50–58, 2014.

[9] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, 2017.

[10] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[11] K. Bauman, B. Liu, and A. Tuzhilin. Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 717–725. ACM, 2017.

[12] P. Biyani, K. Tsioutsiouliklis, and J. Blackmer. " 8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[13] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[14] E. Brill, S. Dumais, and M. Banko. An analysis of the askmsr question-answering system. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 257–264, 2002.

[15] P. Burnap and M. L. Williams. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242, 2015.

[16] P. Burnap and M. L. Williams. Us and them: identifying cyber hate on twitter across multiple protected characteristics. *EPJ Data science*, 5(1):11, 2016.

[17] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[18] A. Chakraborty, B. Paranjape, S. Kakarla, and N. Ganguly. Stop clickbait: Detecting and preventing clickbaits in online news media. In *2016 ieee/acm international conference on advances in social networks analysis and mining (asonam)*, pages 9–16. IEEE, 2016.

[19] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[20] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[21] Z. Chen, N. Ma, and B. Liu. Lifelong learning for sentiment classification. *arXiv preprint arXiv:1801.02808*, 2018.

[22] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014.

[23] Y. Choi and C. Cardie. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 793–801, 2008.

[24] N. Choudhary, R. Singh, I. Bindlish, and M. Shrivastava. Sentiment analysis of code-mixed languages leveraging resource rich languages. *arXiv preprint arXiv:1804.00806*, 2018.

[25] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[26] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[27] T. Cohn and P. Blunsom. Semantic role labelling with tree conditional random fields. 2005.

[28] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, É. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, 2020.

[29] A. Conneau and G. Lample. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7057–7067, 2019.

[30] O. P. Damani. Improving pointwise mutual information (pmi) by incorporating significant co-occurrence. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 20–28, 2013.

[31] A. Das and S. Bandyopadhyay. Sentiwordnet for indian languages. In *Proceedings of the Eighth Workshop on Asian Language Resouces*, pages 56–63, 2010.

[32] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Multilingual bert -r. *https://github.com/google-research/bert/blob/master/multilingual.md*, 2018.

[35] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 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, 2019.

[36] J. C. S. Dos Rieis, F. B. de Souza, P. O. S. V. de Melo, R. O. Prates, H. Kwak, and J. An. Breaking the news: First impressions matter on online news. In *Ninth International AAAI conference on web and social media*, 2015.

[37] M. Dragoni and G. Petrucci. A neural word embeddings approach for multi-domain sentiment analysis. *IEEE Transactions on Affective Computing*, 8(4):457–470, 2017.

[38] P. Ekman. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200, 1992.

[39] A. Esuli and F. Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *LREC*, volume 6, pages 417–422. Citeseer, 2006.

[40] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*, 2017.

[41] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[42] R. R. R. Gangula and R. Mamidi. Resource creation towards automated sentiment analysis in telugu (a low resource language) and integrating multiple domain sources to enhance sentiment prediction. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[43] N. D. Gitari, Z. Zuping, H. Damien, and J. Long. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230, 2015.

[44] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision.

[45] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

[46] C. Hadiwinoto, H. T. Ng, and W. C. Gan. Improved word sense disambiguation using pre-trained contextualized word representations. *arXiv preprint arXiv:1910.00194*, 2019.

[47] T. Hartvigsen, S. Gabriel, H. Palangi, M. Sap, D. Ray, and E. Kamar. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *arXiv preprint arXiv:2203.09509*, 2022.

[48] B. Heinzerling and M. Strube. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018 2018. European Language Resources Association (ELRA).

[49] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[50] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[51] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[52] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[53] M. I. Jordan and L. Xu. Convergence results for the em approach to mixtures of experts architectures. *Neural Networks*, 8(9):1409–1431, 1995.

[54] A. Joshi, P. Bhattacharyya, and M. J. Carman. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)*, 50(5):1–22, 2017.

[55] A. Joshi, V. Tripathi, K. Patel, P. Bhattacharyya, and M. Carman. Are word embedding-based features useful for sarcasm detection? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1006–1011, 2016.

[56] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[57] D. Kakwani, A. Kunchukuttan, S. Golla, N. Gokul, A. Bhattacharyya, M. M. Khapra, and P. Kumar. inlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4948–4961, 2020.

[58] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, 2014.

[59] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

[60] H.-Y. Kim. Analysis of variance (anova) comparing means of more than two groups. *Restorative dentistry & endodontics*, 39(1):74–77, 2014.

[61] S.-M. Kim and E. Hovy. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1367. Association for Computational Linguistics, 2004.

[62] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[63] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[64] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[65] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[66] T. N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[67] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[68] R. Kshirsagar, T. Cukuvac, K. McKeown, and S. McGregor. Predictive embeddings for hate speech detection on twitter. *arXiv preprint arXiv:1809.10644*, 2018.

[69] S. Kumar, A. Kulkarni, M. S. Akhtar, and T. Chakraborty. When did you become so smart, oh wise one?! sarcasm explanation in multi-modal multi-party dialogues. *arXiv preprint arXiv:2203.06419*, 2022.

[70] G. Lample and A. Conneau. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[71] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[72] T. T. Le, W. Fu, and J. H. Moore. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics*, 36(1):250–256, 2020.

[73] Z. Li, F. Tang, M. Zhao, and Y. Zhu. Emocaps: Emotion capsule based model for conversational emotion recognition. *arXiv preprint arXiv:2203.13504*, 2022.

[74] P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2873–2879, 2016.

[75] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[76] Y. Liu, Y. Wang, A. Sun, Z. Zhang, J. Guo, and X. Meng. A dual-channel framework for sarcasm recognition by detecting sentiment conflict. *arXiv preprint arXiv:2109.03587*, 2021.

[77] D. Marcheggiani and I. Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, 2017.

[78] M. Marreddy, S. R. Oota, L. S. Vakada, V. C. Chinni, and R. Mamidi. Clickbait detection in telugu: Overcoming nlp challenges in resource-poor languages using benchmarked techniques. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[79] R. Mihalcea and C. Strapparava. Lyrics, music, and emotions. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 590–599. Association for Computational Linguistics, 2012.

[80] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[81] S. Mohammad and F. Bravo-Marquez. Emotion intensities in tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 65–77, 2017.

[82] S. M. Mohammad, S. Kiritchenko, and X. Zhu. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*, 2013.

[83] S. M. Mohammad and P. D. Turney. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 26–34. Association for Computational Linguistics, 2010.

[84] S. M. Mohammad and P. D. Turney. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465, 2013.

[85] K. Moilanen and S. Pulman. Sentiment composition. 2007.

[86] S. S. Mukku and R. Mamidi. Actsa: Annotated corpus for telugu sentiment analysis. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 54–58, 2017.

[87] S. S. Mukku, S. R. Oota, and R. Mamidi. Tag me a label with multi-arm: Active learning for telugu sentiment analysis. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 355–367. Springer, 2017.

[88] A. Nandy. Beyond words: Pictograms for indian languages.

[89] F. Å. Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*, 2011.

[90] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153, 2016.

[91] S. J. Nowlan and G. E. Hinton. Evaluation of adaptive mixtures of competing experts. In *Advances in neural information processing systems*, pages 774–780, 1991.

[92] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124, 2005.

[93] S. Parupalli, V. A. Rao, and R. Mamidi. Bcsat: A benchmark corpus for sentiment analysis in telugu using word-level annotations. In *Proceedings of ACL 2018, Student Research Workshop*, pages 99–104, 2018.

[94] S. Parupalli and N. Singh. Enrichment of ontosensenet: Adding a sense-annotated telugu lexicon. *arXiv preprint arXiv:1804.02186*, 2018.

[95] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[96] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

[97] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[98] S. Petrović, M. Osborne, and V. Lavrenko. The edinburgh twitter corpus. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, pages 25–26, 2010.

[99] R. Plutchik. The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. *American scientist*, 89(4):344–350, 2001.

[100] A. Poncelas, P. Lohar, A. Way, and J. Hadley. The impact of indirect machine translation on sentiment classification. *arXiv preprint arXiv:2008.11257*, 2020.

[101] M. Potthast, S. Köpsel, B. Stein, and M. Hagen. Clickbait detection. In *European Conference on Information Retrieval*, pages 810–817. Springer, 2016.

[102] Q. Qian, M. Huang, J. Lei, and X. Zhu. Linguistically regularized lstms for sentiment classification. *arXiv preprint arXiv:1611.03949*, 2016.

[103] A. Radford, R. Jozefowicz, and I. Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

[104] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

[105] J. Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

[106] A. Ramponi and S. Tonelli. Features or spurious artifacts? data-centric baselines for fair and robust hate speech detection. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3027–3040. Association for Computational Linguistics, 2022.

[107] A. J. Reagan, L. Mitchell, D. Kiley, C. M. Danforth, and P. S. Dodds. The emotional arcs of stories are dominated by six basic shapes. *EPJ Data Science*, 5(1):31, 2016.

[108] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[109] G. D. Ruxton and G. Beauchamp. Time for some a priori thinking about post hoc testing. *Behavioral ecology*, 19(3):690–693, 2008.

[110] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[111] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[112] R. Sharma, A. Somani, L. Kumar, and P. Bhattacharyya. Sentiment intensity ranking among adjectives using sentiment bearing word embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 547–552, 2017.

[113] W. Shi, F. Li, J. Li, H. Fei, and D. Ji. Effective token graph modeling using a novel labeling strategy for structured sentiment analysis. *arXiv preprint arXiv:2203.10796*, 2022.

[114] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[115] J. Staiano and M. Guerini. Depeche mood: a lexicon for emotion analysis from crowd annotated news. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 427–433, 2014.

[116] S. Stieglitz and L. Dang-Xuan. Emotions and information diffusion in social media—sentiment of microblogs and sharing behavior. *Journal of management information systems*, 29(4):217–248, 2013.

[117] P. J. O. Suárez, B. Sagot, and L. Romary. Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Leibniz-Institut für Deutsche Sprache, 2019.

[118] R. S. Swier and S. Stevenson. Exploiting a verb lexicon in automatic semantic role labelling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 883–890, 2005.

[119] D. Tang, B. Qin, X. Feng, and T. Liu. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307, 2016.

[120] D. Tang, B. Qin, and T. Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015.

[121] K. Tayal, N. Rao, S. Agrawal, and K. Subbian. Short text classification using graph convolutional network. In *NIPS workshop on Graph Representation Learning*, 2019.

[122] R. Tokuhisa, K. Inui, and Y. Matsumoto. Emotion classification using massive examples extracted from the web. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 881–888. Association for Computational Linguistics, 2008.

[123] M. Tummalapalli, M. Chinnakotla, and R. Mamidi. Towards better sentence classification for morphologically rich languages. 2018.

[124] P. D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

[125] B. Varshit, B. V. Vishal, D. M. M. K. Reddy, and R. Mamidi. Sentiment as a prior for movie rating prediction. In *Proceedings of the 2nd International Conference on Innovation in Artificial Intelligence*, pages 148–153, 2018.

[126] H. M. Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984, 2006.

[127] Z. Waseem and D. Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93, 2016.

[128] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics, 2005.

[129] H. Wu and X. Shi. Adversarial soft prompt tuning for cross-domain sentiment analysis. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2438–2447, 2022.

[130] S. Wu and M. Dredze. Are all languages created equal in multilingual bert? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, 2020.

[131] L. Xiao, H. Zhang, and W. Chen. Gated multi-task network for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 726–731, 2018.

[132] C. Xiong, S. Merity, and R. Socher. Dynamic memory networks for visual and textual question answering. In *International conference on machine learning*, pages 2397–2406, 2016.

[133] C. Yang, K. H.-Y. Lin, and H.-H. Chen. Building emotion lexicon from weblog corpora. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 133–136, 2007.

[134] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377, 2019.

[135] S.-J. Yen and Y.-S. Lee. Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In *Intelligent Control and Automation*, pages 731–740. Springer, 2006.

[136] A. Yessenalina, Y. Yue, and C. Cardie. Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1046–1056. Association for Computational Linguistics, 2010.

[137] W. Yin and H. Schütze. Learning word meta-embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1351–1360, 2016.

[138] H.-F. Yu, F.-L. Huang, and C.-J. Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1-2):41–75, 2011.

[139] L.-C. Yu, J. Wang, K. R. Lai, and X. Zhang. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 534–539, 2017.

[140] S. E. Yuksel, J. N. Wilson, and P. D. Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012.

[141] L. Zhang, S. Wang, and B. Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

[142] M. Zhang, Y. Zhang, and G. Fu. Tweet sarcasm detection using deep neural network. In *Proceedings of COLING 2016, The 26th International Conference on Computational Linguistics: Technical Papers*, pages 2449–2460, 2016.

[143] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657, 2015.

[144] Y. Zhang, X. Yu, Z. Cui, S. Wu, Z. Wen, and L. Wang. Every document owns its structure: Inductive text classification via graph neural networks. *arXiv preprint arXiv:2004.13826*, 2020.

[145] Z. Zhang, D. Robinson, and J. Tepper. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European semantic web conference*, pages 745–760. Springer, 2018.

[146] H. Zhao, Z. Lu, and P. Poupart. Self-adaptive hierarchical sentence model. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[147] J. Zhao, M. Liu, L. Gao, Y. Jin, L. Du, H. Zhao, H. Zhang, and G. Haffari. Summpip: Unsupervised multi-document summarization with sentence graph compression. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1949–1952, 2020.

[148] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*, 2016.

[149] Z.-H. Zhou and J. Feng. Deep forest: Towards an alternative to deep neural networks. *arXiv preprint arXiv:1702.08835*, 2017.

[150] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.