Graph Convolutional Networks with Multi-headed Attention for Code-Mixed Sentiment Analysis

by

Suman Dowlaqar, Radhika Mamidi

Report No: IIIT/TR/2021/-1

Centre for Language Technologies Research Centre
International Institute of Information Technology
Hyderabad - 500 032, INDIA
April 2021

Graph Convolutional Networks with Multi-headed Attention for Code-Mixed Sentiment Analysis

Suman Dowlagar
LTRC
IIIT-Hyderabad
suman.dowlagar
@research.iiit.ac.in

Radhika Mamidi
LTRC
IIIT-Hyderabad
radhika.mamidi
@iiit.ac.in

Abstract

Code-mixing is a frequently observed phenomenon in multilingual communities where a speaker uses multiple languages in an utterance or sentence. Code-mixed texts are abundant, especially in social media, and pose a problem for NLP tools as they are typically trained on monolingual corpora. Recently, finding the sentiment from code-mixed text has been attempted by some researchers in SentiMix SemEval 2020 and Dravidian-CodeMix FIRE 2020 shared tasks. Mostly, the attempts include traditional methods, long short term memory, convolutional neural networks, and transformer models for code-mixed sentiment analysis (CMSA). However, no study has explored graph convolutional neural networks on CMSA. In this paper, we propose the graph convolutional networks (GCN) for sentiment analysis on code-mixed text. We have used the datasets from the Dravidian-CodeMix FIRE 2020. Our experimental results on multiple CMSA datasets demonstrate that the GCN with multi-headed attention model has shown an improvement in classification metrics.

1 Introduction

Through the advent of social media, people from around the world can connect and exchange information instantly. The evolution of social media texts from blogs and messaging websites has created many new opportunities for information access worldwide (Thavareesan and Mahesan, 2019, 2020a,b). Multilingual communities often express their thoughts on social media by mixing different languages in the same utterance (Chakravarthi, 2020a). This mixing or alteration of two or more languages is known as code-mixing or code-switching (Wardhaugh, 2011; Chakravarthi, 2020b).

The computational modeling of code-mixed text is challenging due to the linguistic complexity, nature of mixing, the presence of non-standard variations in spellings, grammar, and transliteration (Bali et al., 2014; Jose et al., 2020; Priyadharshini et al., 2020). Because of such non-standard variations, CM poses several unseen difficulties in fundamental fields of natural language processing (NLP) tasks such as language identification, part-of-speech tagging, shallow parsing, Natural language understanding, sentiment analysis, etc.

To encourage research on code-mixing and to solve the problems related to code-mixed text, the NLP community organizes several tasks and workshops such as Task9: SentiMix, SemEval 2020 (Patwa et al., 2020), and 4th Workshop on Computational Approaches for Linguistic Code-Switching (Solorio et al., 2020), Dravidian Code-Mix FIRE 2020 (Chakravarthi et al., 2020c; Mandl et al., 2020). All of these tasks focus on sentiment analysis on code-mixed text.

Some researchers used traditional methods, such as support vector machines (SVM's) (Vapnik, 2013) and Logistic Regression (LR), where the code-mixed text is represented with sparse linguistic features (e.g., bag-of-words and n-grams). Others used deep learning models such as convolutional neural networks (CNN) (Kim, 2014), recurrent neural networks (RNN) (Hochreiter and Schmidhuber, 1997), transformer models (Vaswani et al., 2017), etc. The above deep learning models can capture semantic and syntactic information well in local consecutive word sequences. Such position-dependent models are not suitable for Dravidian languages that follow free word order format.

Recently, an evolving research direction known as graph neural networks (Zhou et al., 2018) or graph representation learning has attracted wide attention. In graph-based approaches, the whole corpus is viewed as a graph (Yao et al., 2019).

Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, pages 65–72

Graph convolutional networks have been effective at tasks for knowledge representation and can preserve global structure information of a graph in graph embeddings.

In this work, we propose the use of the graph neural network for code-mixed sentiment analysis. Our method focuses on constructing a single large graph from an entire corpus. We modeled the graph with a Graph Convolutional Network (GCN) (Kipf and Welling, 2016; Yao et al., 2019) with Multi-headed attention. A graph neural network captures high-order neighborhood information well by using an adjacency matrix. This adjacency matrix helps the graph neural network to not dependent on consecutive word order format. Thus, it is well suited for the Dravidian language datasets that follow a free word order.

The paper is organized as follows. Section 2 provides related work on code-mixed sentiment analysis. Section 3 describes the proposed work. Section 4 presents the experimental setup and the performance of the model. Section 5 concludes our work.

2 Related work

In this section, we study the related work in the field of code-mixed sentiment analysis.

2.1 Traditional methods for CMSA

Traditional methods for CMSA studies mainly focus on feature engineering and traditional text classification algorithms such as Support Vector Machines (SVMs), Decision Trees (DTs), Logistic Regression (LR) (Chakravarthi et al., 2020c). The most commonly used feature repreentation is the bag-of-words. Some linguistic and complex features have been designed to capture the features, such as n-grams, chunking, capturing phonetics of words, etc.

2.2 Deep Learning for CMSA

Deep learning studies focus on using deep neural network classifiers for CMSA. Kumar et al. (2020) used CNN for code-mixed sentiment analysis. Mahata et al. (2020); Lakshmanan and Ravindranath (2020) used LSTM, a specific type of Recurrent neural network, to learn text representation. To further increase the representation ability of neural network models, attention mechanisms, and transformer models have been introduced for code-switched sentiment analysis (Chakravarthi et al., 2020c). These methods mainly focus on the position of the word and local consecutive word sequences and do not explicitly model global word co-occurrence information in a corpus.

2.3 Graph Neural Networks

Recently, Graph Neural Networks has received growing attention in the field of natural language processing. The authors used Text GCN (Yao et al., 2019) to construct a heterogeneous graph that helped them in text classification and achieved great results. Graph Neural Networks have also been used in intent slot labeling (Zhang et al., 2020), semantic role labeling (Marcheggiani and Titov, 2017), knowledge base construction (Wang et al., 2019), relation classification (Sahu et al., 2019), and machine translation (Marcheggiani et al., 2018).

The use of graph convolutional networks with undirected graphs to model the free word order languages for CMSA has never been explored before.

3 Proposed Work

In this section, we describe our proposed work. First, we present the pre-processing steps to filter the data, and then we describe graph-convolutional networks with multi-headed attention for code-mixed sentiment analysis.

The data we have used for CMSA consists of code-mixed Tamil-English and Malayalam-English youtube comments, and it has many irregularities. Such as the Roman script is used to write Tamil and Malayalam text with variations in the spelling, irregular grammatical structure, mixing of languages at the word level, informal abbreviations, slang, and use of emoticons in text. We can minimize some of these irregularities by using pre-processing methods.

3.1 Pre-processing

In pre-processing

- We tokenized the text on whitespaces and symbols, including colons, commas, and semicolons. It was followed by the removal of hashtags, URLs, mentions, numbers.

- In the sentiment analysis scenarios, emoticons help in understanding the polarity of the text. We used the emoji PyPI library to map the emoticons to their corresponding text.

- We know that the code-mixed text document contains multiple scripts involving multiple languages. Each language may or may not use its own native script. When a word is represented in a different script, spelling variations can occur across document(s). This will create unnecessary complexity for the classifier to learn a meaningful representation of a word (that is written in two or more languages in the same document), which results in poor performance of the classifier. In order to resolve such ambiguities, it is necessary to bring the words to a common format. i.e., back transliteration of each word in its native script. We have used AI4Bharat-Transliteration[1] (a python deep transliteration engine for transliteration from roman to native text).

3.2 Word embeddings

After transliteration, we have used the pre-trained fastText model (Grave et al., 2018) to obtain the word embeddings. We didn't rely on extracting word-embeddings from the same corpora as its size is small, and we need a large code mixed corpora with cross-lingual mapping or cross-lingual dictionary to obtain better word-embeddings. Even Khanuja et al. (2020) has suggested that we can use pre-trained models to learn representations of words. Their paper has shown an improved performance in accuracy when pre-trained models are used on the code-mixed datasets.

3.3 Graph Convolutional Networks (GCN) with Multi-headed Attention

A Graph Convolutional Network is a neural network that operates on graph-like representations. Given a graph with V as the number of vertices and E as the number of edges $G = (V, E)$,

A GCN takes as input an input feature matrix X of size (N, F^0) feature matrix, where N is the number of nodes or vertices, and F^0 is the number of input features or node embeddings for each node, and An adjacency matrix A for graph G of size (N, N). A Degree matrix D of size $(1, N)$. A hidden layer in the GCN can be defined as $H^i = f(H^{i-1}, A))$ where $H^0 = X$ and f is a propagation rule. Each layer H^i corresponds to an (N, F^i) feature matrix where each row is a feature representation of a node. The neighboring features are aggregated at each layer to form the next layer's features by using the propagation rule f. In this way, features become increasingly more abstract at each consecutive layer.

A simple propagation rule in GCN is given as:

$$f(H^i, A) = \sigma(AH^iW^i) \tag{1}$$

Where W^i is the weight matrix for layer i, and σ is a non-linear activation function. The weight matrix has dimensions (F^i, F^{i+1}). Equation 1 can be explained as a feature transformation that occurs with information borrowed from the neighborhood nodes. But the transformation in the equation 1 focuses only on neighborhood information and ignores its own node features, so we add the identity matrix to the adjacency matrix. Later we normalize the feature representations w.r.t degree of each node. So the initial propagation rule is modified as given in equation 2

$$f(H^i, A) = \sigma(\tilde{A}H^iW^i) \tag{2}$$

where $\tilde{A} = D^{-1/2}(A + I)D^{-1/2}$ is the normalized symmetric adjacency matrix.

The multiheaded attention is applied on the Hidden layer representation of each graph convolutional layer. The multiheaded attention is given by,

$$Attention = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right).V \tag{3}$$

Multi-headed attention gives high weightage to the words of importance for a given query word in a sentence.

3.4 GCN with Multi-headed Attention on Text

As mentioned earlier, we use graph-neural networks for code-mixed sentiment analysis. Our approach considers the whole corpus as a graph. Then we perform graph convolutions for CMSA.

In order to construct the whole text as a graph, we use the approach given in the paper Yao et al. (2019). Initially, we build a large corpus graph that contains each word and each document as nodes. The number of nodes or vertices in the text graph $|V|$ is the number of documents plus the number of unique words. We build edges based on word occurrence in documents(document-to-word) and word co-occurrence (word-to-word) in the whole corpus. The weights of edges between word-to-word and document-to-word are given as

[1] https://pypi.org/project/ai4bharat-transliteration/

- **Document-to-word:** The weight of the edge between a document node and a word node is the term frequency-inverse document frequency (TF-IDF) measure of the word in the document.

- **Word-to-word:** We use point-wise mutual information (PMI), to calculate weights between two nodes. A positive PMI value indicates a high semantic correlation of words in a corpus, whereas a negative PMI value indicates little or no semantic correlation in the corpus. As suggested in the paper, we only add edges between word pairs with positive PMI values. Note: There are no direct document-to-document relations in the graph.

Adjacency matrix,

$$A_{ij} = \begin{cases} PMI(i,j) & i, j \text{ are words} \\ TF-IDF_{i,j} & i \text{ is document, } j \text{ is word} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $PMI(i,j) > 0$ and PMI value of a word pair i, j is computed as

$$PMI(i,j) = log\frac{p(i,j)}{p(i)p(j)} \tag{5}$$

where $p(i,j)$ is the probability of sliding window occurences that contains both the words i and j. $p(i)$ and $p(j)$ are the individual probability distribution of words i and j respectively.

And Identity matrix is given as,

$$I_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

After building the global text graphs, we apply GCN with multilayer GCN as described above. We used a 3-layer of GCN with multi-headed attention, followed by a global max pooling and then a linear layer to obtain the code-mixed text's polarity. A three-layer GCN can allow message passing among neighborhood nodes that are at a maximum of three steps away. This approach resulted in obtaining a better graph representation.

4 Experiments

In this section, we compare our method with several baselines. We used only the back transliterated dataset to run all the models.

- **TF-IDF + SVM:** A bag-of-words model with term frequency-inverse document frequency measure. Support Vector machines (SVM) is used as the classifier.

- **CNN:** A Convolutional Neural Network model for CMSA. The CNN model uses pre-trained fastText word embeddings.

- **LSTM:** The LSTM model uses the last hidden state to represent the whole sentence. We used fastText pre-trained word embeddings.

- **ELMO + SVM:** Deep Contextualised Word Representations (Peters et al., 2018) uses a deep, bi-directional LSTM model to create word representations. ELMo obtained the word representations by analyzing words within the context that they are used. It also uses character-based embeddings to model the representations for out-of-vocabulary words.

- **BERT(pre-trained multilingual)**: We used pre-trained multilingual bi-directional encoder representations using transformers (BERT) (Devlin et al., 2018) for CMSA.

- **TextGCN:** It is a graph convolutional network implemented for text classification.

4.1 Dataset

For Dravidian code-mixed sentiment analysis, we used the dataset provided by the organizers of Dravidian Code-mixed FIRE-2020. The dataset consists of 15,744 Tamil-English Code-mixed and 6,739 Malayalam-English Code-mixed youtube video comments. The data statistics are given in table 1. Each sentence's sentiment can be classified into five categories: positive, negative, not-Tamil or not-Malayalam, unknown_state, mixed_feelings. The dataset details and the initial benchmarks on the corpus are given in Chakravarthi et al. (2020a,b).

4.2 Model parameters and settings

For GCN with multi-headed attention model, we set the embedding size to 300. The embeddings are obtained by using fastText pre-trained models. We used adam optimizer with cross-entropy loss. We tuned other parameters and set the learning rate as 0.02, dropout as 0.25, and we trained the GCN for a maximum of 100 epochs. We stopped the model if the validation loss does not decrease for

Dataset	#Docs	#Train	#Dev	#Test	#classes
Tamil-English CM	15744	11335	1260	3149	5
Malayalam-English CM	6739	4851	540	1348	5

Table 1: Data Statistics

Model	Pos	Neg	not-mal	Mixed	Unknown	Macro	Weight	Acc
TF-IDF+SVM	0.72	0.40	0.98	0.27	0.58	0.59	0.68	0.67
CNN	0.76	0.39	0.93	0.23	0.63	0.59	0.68	0.69
LSTM	0.74	0.44	0.93	0.26	0.63	0.60	0.68	0.69
ELMO+SVM	0.75	0.45	0.96	0.26	0.63	0.61	0.69	0.69
mBERT	0.77	0.53	**0.99**	0.26	**0.69**	0.65	0.72	0.72
Text-GCN	0.79	**0.55**	0.98	**0.30**	0.67	**0.66**	0.74	**0.73**
Our approach	**0.80**	**0.55**	**0.99**	**0.30**	**0.69**	**0.66**	**0.75**	**0.73**

Table 2: Classification Metrics of our GCN with multi-headed approach when compared to baselines on code-mixed Malayalam-English Dataset

Model	Pos	Neg	not-Tamil	Mixed	Unknown	Macro	Weight	Acc
TF-IDF+SVM	0.80	0.19	0.93	0.06	0.12	0.42	0.56	0.68
CNN	**0.81**	0.04	0.82	0.01	0.00	0.34	0.56	0.68
LSTM	**0.81**	0.16	0.85	0.01	0.15	0.40	0.59	0.68
ELMO+SVM	**0.81**	0.17	0.88	0.04	0.15	0.43	0.58	0.69
mBERT	0.77	**0.53**	**0.99**	**0.26**	**0.69**	**0.65**	**0.72**	**0.72**
Text-GCN	**0.81**	0.30	0.98	0.05	0.02	0.43	0.61	0.70
Our approach	**0.81**	0.43	0.98	0.16	0.15	0.45	0.64	0.71

Table 3: Classification Metrics of our GCN with multi-headed approach when compared to baselines on code-mixed Tamil-English Dataset

ten consecutive epochs. We used default parameter settings for baseline models as given in their original papers or implementations and used the same pre-trained embeddings obtained from the fastText model. We ran the model more than ten times, and the average metrics are recorded. For baseline models, we used default parameter settings as in their original papers or implementations. For baseline models using pre-trained word embeddings, we used 300-dimensional FastText word embeddings.

We used pytorch-geometric[2] library to implement the graph convolutional networks. Pytorch-geometric library has a set of predefined graph neural network models that can be accessed by calling a simple method. We have used the hugging face transformers library (Wolf et al., 2019) to download the pre-trained multilingual BERT model. The scikit-learn library (Pedregosa et al., 2011) is used to obtain the classification metrics.

[2] https://pytorch-geometric.readthedocs.io/en/latest/index.html

4.3 Results and Analysis

Table 3 and 2 presents F1-scores and accuracy of each model on Tamil-English and Malayalam-English datasets respectively. For the Malayalam-English dataset, GCN with Multi-headed attention outperforms all baseline models. It showcases the proposed method's effectiveness on short text datasets. For the Tamil dataset, our model performed competitively when compared to the pre-trained multilingual BERT model and better than all the other baselines. When compared to the BERT introduced in the original paper (Chakravarthi et al., 2020b), our GCN with multi-headed attention achieved better results. We have seen improvement in results in our pre-trained multilingual BERT because we used back transliteration while pre-processing. The back transliteration helped the BERT pre-trained model captured the native word script, thus obtaining better pre-trained sub-word level embeddings. Our model performed better than the Text-GCN because of multi-headed attention. CNN also didn't perform

well on both the datasets when compared to the baselines. It might be because there are long-range and implicit dependencies between the text, but CNN only works on exploring the short-range dependencies. LSTM-based models are good at establishing long-range dependencies in the datasets. This helped the LSTM model to perform better when compared to CNN models.

The pre-trained multilingual BERT model has shown a decrease in performance on "positive" polarity of the Tamil-English CM dataset because the model tried to fine-tune the data on other polarities. The original paper reported that there were annotating issues for mixed and unknown sentences as their polarity was similar to the "positive" sentence. Fine-tuning the classifier towards other polarities resulted in a decrease in the classification metrics of the "positive" polarity sentences.

In all the baseline models, We have seen an increase in accuracy compared to the baseline papers on the above datasets (Chakravarthi et al., 2020a,b). It is due to the back-transliteration module used while pre-processing. The back transliteration helped the words retain their native script and helped the words obtain better-pre-trained embeddings from the fastText classification models, and decreased the classifier's training complexity. Especially the accuracy is improved for the "Positive" and "not-tamil/not-malayalam" class. The "not-tamil"/"not-malayalam" class has shown an improved accuracy because we have formulated rules not to transliterate these words into the native script. It helped the classifier to understand them as out-of-context words while training the classifier. Similar to the baseline models, the negative, mixed_feelings, and unknown_state have shown low performance because of their low distribution of data compared to the positive class.

Our graph convolution network with attention mechanism showed good results because

- The weighted label information of document nodes is passed to their neighboring word nodes, then transferred to other word nodes and document nodes that are neighbor to the first step neighboring word nodes. This knowledge transfer helped the graph gather comprehensive word-document information, which helped in learning a better classifier for the given data.
- Using back transliteration with pre-trained fastText word embeddings preserved syntactic and semantic relations among words and provided additional information in classifying the data.

After data analysis, we observed that the negative polarity comments have sarcasm included. The words used in the sarcasm are similar to those of positive comments. It made the classifier identify them as positive words and made sentiment analysis a difficult task. Mixed feelings had both positive and negative sentences. As the classifier was trained on a lot of positive instances, the mixed feelings were tuned to the "positive" class. And the classifier mislabeled them as a "positive" class. It affected the performance of the classifier. More training data or additional labeling of the classifier could help resolve such issues.

5 Conclusion

This paper describes the graph convolution networks with multi-headed attention for code mixed sentiment analysis on Dravidian languages. Initially, we transliterate the data and build a word document graph for the whole corpus. This approach considers global dependencies between words and documents in the corpus. A simple three-layer GCN with Multi-headed Attention on CMSA demonstrates encouraging results by outperforming numerous state-of-the-art methods. Our approach obtained the best results on the Malayalam-English CM dataset with a weighted-F1 score and an accuracy of 0.75 and 0.73. We will explore our model's performance on various graph networks with different filtering and smoothing techniques for future work.

References

Kalika Bali, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas. 2014. "I am borrowing ya mixing?" An Analysis of English-Hindi Code Mixing in Facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 116–126.

Bharathi Raja Chakravarthi. 2020a. HopeEDI: A multilingual hope speech detection dataset for equality, diversity, and inclusion. In *Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media*, pages 41–53, Barcelona, Spain (Online). Association for Computational Linguistics.

Bharathi Raja Chakravarthi. 2020b. *Leveraging orthographic information to improve machine translation*

of under-resourced languages. Ph.D. thesis, NUI Galway.

Bharathi Raja Chakravarthi, Navya Jose, Shardul Suryawanshi, Elizabeth Sherly, and John Philip McCrae. 2020a. A sentiment analysis dataset for code-mixed Malayalam-English. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 177–184, Marseille, France. European Language Resources association.

Bharathi Raja Chakravarthi, Vigneshwaran Muralidaran, Ruba Priyadharshini, and John Philip McCrae. 2020b. Corpus creation for sentiment analysis in code-mixed Tamil-English text. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 202–210, Marseille, France. European Language Resources association.

Bharathi Raja Chakravarthi, Ruba Priyadharshini, Vigneshwaran Muralidaran, Shardul Suryawanshi, Navya Jose, Elizabeth Sherly, and John P. McCrae. 2020c. Overview of the Track on Sentiment Analysis for Dravidian Languages in Code-Mixed Text. In *Forum for Information Retrieval Evaluation*, FIRE 2020, page 21–24, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. *arXiv preprint arXiv:1802.06893*.

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

Navya Jose, Bharathi Raja Chakravarthi, Shardul Suryawanshi, Elizabeth Sherly, and John P. McCrae. 2020. A Survey of Current Datasets for Code-Switching Research. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 136–141.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. GLUECoS: An Evaluation Benchmark for Code-Switched NLP. *arXiv preprint arXiv:2004.12376*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Ayush Kumar, Harsh Agarwal, Keshav Bansal, and Ashutosh Modi. 2020. BAKSA at SemEval-2020 Task 9: Bolstering cnn with self-attention for sentiment analysis of code mixed text. *arXiv preprint arXiv:2007.10819*.

BalaSundaraRaman Lakshmanan and Sanjeeth Kumar Ravindranath. 2020. Theedhum Nandrum@ Dravidian-CodeMix-FIRE2020: ASentiment Polarity Classifier for YouTube Commentswith Code-switching between Tamil, Malayalam and English. *arXiv preprint arXiv:2010.03189*.

Sainik Kumar Mahata, Dipankar Das, and Sivaji Bandyopadhyay. 2020. JUNLP@ Dravidian-CodeMix-FIRE2020: Sentiment Classification of Code-Mixed Tweets using Bi-Directional RNN and Language Tags. *arXiv preprint arXiv:2010.10111*.

Thomas Mandl, Sandip Modha, Anand Kumar M, and Bharathi Raja Chakravarthi. 2020. Overview of the HASOC Track at FIRE 2020: Hate Speech and Offensive Language Identification in Tamil, Malayalam, Hindi, English and German. In *Forum for Information Retrieval Evaluation*, FIRE 2020, page 29–32, New York, NY, USA. Association for Computing Machinery.

Diego Marcheggiani, Joost Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. *arXiv preprint arXiv:1804.08313*.

Diego Marcheggiani and Ivan Titov. 2017. 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, Copenhagen, Denmark. Association for Computational Linguistics.

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas PYKL, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. Semeval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. *arXiv preprint arXiv:2008.04277*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Ruba Priyadharshini, Bharathi Raja Chakravarthi, Mani Vegupatti, and John P. McCrae. 2020. Named Entity Recognition for Code-Mixed Indian Corpus using Meta Embedding. In *2020 6th International*

Conference on Advanced Computing and Communication Systems (ICACCS), pages 68–72.

Sunil Kumar Sahu, Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. Inter-sentence relation extraction with document-level graph convolutional neural network. *arXiv preprint arXiv:1906.04684*.

Thamar Solorio, Monojit Choudhury, Kalika Bali, Sunayana Sitaram, Amitava Das, and Mona Diab, editors. 2020. *Proceedings of the The 4th Workshop on Computational Approaches to Code Switching*. European Language Resources Association, Marseille, France.

Sajeetha Thavareesan and Sinnathamby Mahesan. 2019. Sentiment Analysis in Tamil Texts: A Study on Machine Learning Techniques and Feature Representation. In *2019 14th Conference on Industrial and Information Systems (ICIIS)*, pages 320–325.

Sajeetha Thavareesan and Sinnathamby Mahesan. 2020a. Sentiment Lexicon Expansion using Word2vec and fastText for Sentiment Prediction in Tamil texts. In *2020 Moratuwa Engineering Research Conference (MERCon)*, pages 272–276.

Sajeetha Thavareesan and Sinnathamby Mahesan. 2020b. Word embedding-based Part of Speech tagging in Tamil texts. In *2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)*, pages 478–482.

Vladimir Vapnik. 2013. *The nature of statistical learning theory*. Springer science & business media.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Qingyun Wang, Lifu Huang, Zhiying Jiang, Kevin Knight, Heng Ji, Mohit Bansal, and Yi Luan. 2019. Paperrobot: Incremental draft generation of scientific ideas. *arXiv preprint arXiv:1905.07870*.

Ronald Wardhaugh. 2011. *An introduction to sociolinguistics*, volume 28. John Wiley & Sons.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Linhao Zhang, Dehong Ma, Xiaodong Zhang, Xiaohui Yan, and Houfeng Wang. 2020. Graph LSTM with Context-Gated Mechanism for Spoken Language Understanding. In *AAAI*, pages 9539–9546.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*.