

Ajees@HASOC-Dravidian-CodeMix-FIRE2020

Ajees A P

Assistant Professor, Department of Information Technology, Kannur University, Kerala, India

Abstract

There is a dramatic increase in the use of social media over the last decade. People use them for different purposes such as chatting with friends, sharing thoughts, express emotions, etc. But most of such texts are code-mixed and there is no guarantee that they only contain polite words. Here comes the increasing demand for offensive language detection from code-mixed social media texts. Identifying offensive messages from social media texts is a key challenge in social media text processing. The offensive words used in code-mixed text plays a vital role in detecting offensive messages from social media text. In this paper, different representations of code-mixed text are experimented to detect the offensive content in social media text. Along with different word representations, different machine learning architectures have also experimented on the same dataset. The experiments were conducted using the datasets provided by HASOC-Dravidian-Codemix@FIRE2020. The results demonstrate that the proposed systems are capable of giving a comparable performance to the methods employing more sophisticated approaches. Hence, it was able to secure 8th position in the final evaluation phase for the task2.

Keywords

Offensive language detection, MLP, BiLSTM, BERT, CountVectorizer

1. Introduction

Offensive language detection is the process of distinguishing offensive posts and non-offensive posts in social media text. It is a well-known task that finds important applications in fields like social media text analysis, authorship profiling, offensive text removal, etc. Offensive languages should be removed from any publicly available text. Nowadays, due to the increased usage of social media sites and online interactions, getting an offensive message is a common issue faced by commuters. If a comment or post induces any type of offensive language, then recognizing such comments will be one of the crucial measures in finding the source. Speakers of different languages may make different types of offensive messages when creating a post or comment on social media and online news channels. The use of abusive language can even corrupt chatbots. Since chatbots learn from communication with humans if it is not able to differentiate abusive and non-abusive content then it may also use it. Hence offensive language detection finds its huge responsibility for the government as well as social media sites to detect this hate speech content. As a result, several research works [1, 2, 3, 4, 5] across the world have been reported over the past few years to detect the abusive text content in social media sites.

For a country like India, most of the people use their regional language for chatting, commenting, and tweeting. But most of such text is code-mixed. Code-mixing is a common phenomenon in a multilingual society [6, 7]. The use of non-native scripts is another property of social media texts which makes offensive language identification a bit harder. The approaches that only depend on monolingual data usually fail on code-mixed data due to the frequent code-switching behavior observed in social


FIRE 2020: Forum for Information Retrieval Evaluation, December 16-20, 2020, Hyderabad, India

EMAIL: ajeesap87@gmail.com (A.A. P)

ORCID: 0000-0002-4960-1865 (A.A. P)



© 2020 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

media texts. Hate speech detection can't be addressed simply by filtering of abusive words. Along with the meaning of words, various other factors such as characteristics of the user, context information, and the gender of individual people have to be considered for detecting Hate Speech. Abuse is a term that comprises varying forms of fine-grained adverse expressions in the context of natural language. The Hate Speech and Offensive Content Identification in Indo-European Languages (HASOC) organizer team describes Hate Speech as describing negative attributes or deficiencies toward groups because of race, political opinion, sexual orientation, gender, social status, health condition or similar [8, 9]. We participated in Dravidian language offensive language identification sub track. Tamil and Malayalam are Dravidian languages [10, 11, 12, 13]

2. Related Work

Social media plays an important role in communication between people from different demographic groups. With the exponential rise in the usage of hand-held electronic devices across the globe, people started spending huge amounts of time on social media platforms like Twitter, Facebook, and Instagram. Previous studies [14] indicate that most of the online text generated on these platforms contains a considerable amount of abusive language. Cyber terrorism and cyber trolling have become a major threat to the human community. As social media and online platforms have achieved great acceptance and impact among its users, various issues such as Hate Speech, Offensive language, and Profanity have increased drastically on these platforms. Various systems have been proposed by researchers across the world for automatic detection and classification of hate speech.

Shervin et.al [15] proposed a model using character n-grams, word n-grams, and word skip-grams for the classification of English tweets to hate speech, offensive, and non-offensive content. They employed SVM as the classifier with a performance of 78% (accuracy). Recurrent neural networks were applied by Georgious et.al [16] to detect hateful content in social media. Various features like users' tendency towards racism or sexism, user characteristics, etc were also fed to the network. A publicly available corpus with 16000 tweets was used for experiments. Another approach using domain-specific word embeddings was proposed by Satyajit et.al in 2019 [17]. They collected around 25000 tweets using Twitter API and trained a word2vec model to generate domain-specific word embeddings. Code-mixed text containing around 4500 data points were used for training. Modern machine learning techniques such as CNN, LSTM, and BiLSTM were used for classification. Mishra et.al [18] used BERT embeddings towards the feature extraction phase. 'bert-base-multilingual-uncased' model was used for extracting the embedding vectors for the Hindi language. They were able to reach within 1% of the best solution for 5 out of the 8 sub-tasks in the competition.

3. Task Description and Dataset Details

The goal of this shared task is to identify the posts/comments containing offensive language in the Dravidian code-mixed dataset collected from social media [19, 20, 21, 22]. The task is divided into two parts say task1 and task2. Task1 is a message level label classification task focused on YouTube comments. Here the objective is to classify the given YouTube comments in code-mixed Malayalam form as offensive or not-offensive. Task2 is also a message level label classification task focused on Twitter data. The dataset for task2 contains two types of comments namely Tanglish and Manglish (Tamil and Malayalam written using Roman characters). Here also the objective is to classify the code-mixed text in Tamil and Malayalam as offensive or non-offensive. The comment/post in the dataset may contain more than one sentence but the average length of the sentence in the corpora

is one. Each comment/post in the training set is annotated with offensive/not offensive labels at the comment/post level. This dataset is not free from class imbalance problems depicting the real-world scenarios. As per the organizer's claim, this is the first shared task on offensive language detection in Dravidian code-mixed text.

4. Proposed Method

Mainly three types of machine learning models were experimented using two types of word embedding techniques. The first architecture was a simple MLP classifier using two hidden layers. The second one was a combination of CNN-BiLSTM network with four convolutional layers. And the third one was a BiLSTM stack with two hidden layers. In order to represent the individual words in the comments, a simple CountVectorizer as well as BERT was used. CountVectorizer is used to convert text data to a vector of token counts. It also provides the pre-processing of text data before generating the vector representation. This capability makes it highly flexible towards feature representation for text processing. On the other hand, BERT is a pre-trained NLP model that provides contextualized word embeddings [23]. Static word embedding techniques provide the same vector for polysemous words without proper consideration of their context. Whereas the dynamic embedding techniques like ELMo and BERT can consider the context of words before generating their embeddings. BERT word representations take the entire input sentence into the equation for calculating the word embeddings.

4.1. Neural Network based approach

Here the problem is shaped as a text classification task with offensive/not offensive names as labels (classes). The number of classes is equal to the number of labels considered in the study. Text from each post in the training data is considered as a training sample. Providing sequences of raw words will make no sense to the machines. Hence, the complete training set is converted into numeric values using CountVectorizer, a python functionality. The labels of each post are converted to zero and one depending on the label. Finally, this vector representation is provided to an MLP classifier with two hidden layers¹. Different configurations of the neural network with different hidden sizes have experimented. A network with (2,200) hidden configuration was performing well as compared with the other configurations, where 2 is the number of hidden layers and 200 is the hidden layer size.

4.2. Deep Learning based approach

The second model employed a CNN-BiLSTM architecture which used convolutional filters of various sizes. Here the words in the training data were first converted into its corresponding index using a dictionary. Then the training data was zero-padded to make them uniform in length. The sequential model of Keras was used for model construction [24]. The network was configured with four convolutional layers, two max-pooling layers, one dense layer, and an embedding layer. The embedding layer is the first layer that performs the word embeddings. The embedding size was finalized as 50. The second layer is a convolutional layer for its ability to capture the local context. The following layers (max-pooling and convolutional) were arranged in an alternate pattern for acquiring the patterns within the comments/posts. 'Relu' was used as the activation function to bring nonlinearity. The configuration of the CNN part of the network is (7, 7, 3, 3) and (756,756,256,256), where the first four is

¹<https://github.com/ajeesh/HASOC>

Table 1

Results of MLP with CountVectorizer based feature representation on task1

Hidden layers	Hidden size	epochs	Accuracy(%)
2	100	20	93.25
2	200	20	93.5
2	400	20	93.75
2	800	20	93.75
2	400	10	93
3	400	20	93.5
3	400	20	93.5

Table 2

Results of BiLSTM with BERT based feature representation on task1

Hidden layers	Hidden size	feature size (word)	Accuracy(%)
2	300	768	92.5
2	400	768	91.25
3	300	768	92.5
3	400	768	92.5
4	300	768	92.5

the kernel size of each convolutional layer and the second four indicate the corresponding filters. The convolutional layers were followed by a BiLSTM stack containing 400 neurons. The BiLSTM stack is followed by a time distributed dense layer with ‘ReLU’ activation. And the final layer is a dense layer with softmax activation units.

The third architecture was a BiLSTM stack followed by dense layers. The hidden layers contain 300 neurons with ‘Tanh’ activation followed by a dense layer of 100 neurons. The final layer is also a dense layer with a softmax activation unit. Here the input layer accepts the BERT embeddings as input features. ‘bert-base-multilingual-uncased’ model was used for extracting the embedding vectors from the code-mixed text. The embedding size was 768. Each word in the comment/post was replaced by the sum of the vectors from the last four hidden states returned by the BERT model.

5. Results

Different experiments were conducted using different configurations of the neural network models. Table 1 shows the results of CountVectorizer based feature representation on MLP classifier (task1). Similarly table 2 shows the results of BERT based feature representation on the BiLSTM stack. Table 3 shows the impact of the CNN-BiLSTM combination on the same dataset. Similar kind of experiments was also conducted for task2. All the given results are based on the validation dataset provided in the competition. Due to space constraints, the results of those experiments are not presented here. Even though comparable results were obtained for the pre-evaluation phase (on development dataset) of both the tasks, the final results of task1 experiments were not promising. I strongly believe that it may be due to some formatting issues related to the final submission files. The significant difference between the validation performance and test performance ensure this point.

Table 3

Results of CNN-BiLSTM stack on task1

Convolutional layers	filter size	BiLSTM layers	Hidden size	Accuracy(%)
2	(256,256)	2	300	91.5
2	(756,756)	2	300	92
4	(756,756,256,256)	2	300	92.5
4	(756,756,256,256)	2	400	92.5
4	(756,756,256,256)	3	300	92.5

6. Conclusion

I have presented my naive approach towards classifying social media posts in Dravidian-code-mixed text, for hate-speech and offensive content. No external resources like collection of abusive words, negative social media comments, code-mixed abusive text, etc. were used for experiments. Results are only based on the training and development data and expected to improve through the application of modern NLP techniques. More improvements to the results can also be made by incorporating external resources like collection of abusive words, code-mixed abusive text, negative social media comments, etc as additional features.

References

- [1] P. Badjatiya, S. Gupta, M. Gupta, V. Varma, Deep learning for hate speech detection in tweets, in: Proceedings of the 26th International Conference on World Wide Web Companion, 2017, pp. 759–760.
- [2] A. Bohra, D. Vijay, V. Singh, S. S. Akhtar, M. Shrivastava, A dataset of Hindi-English code-mixed social media text for hate speech detection, in: Proceedings of the second workshop on computational modeling of people’s opinions, personality, and emotions in social media, 2018, pp. 36–41.
- [3] J. Chen, S. Yan, K.-C. Wong, Verbal aggression detection on twitter comments: Convolutional neural network for short-text sentiment analysis, *Neural Computing and Applications* (2018) 1–10.
- [4] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, Y. Chang, Abusive language detection in online user content, in: Proceedings of the 25th international conference on world wide web, 2016, pp. 145–153.
- [5] Z. Waseem, D. Hovy, Hateful symbols or hateful people? predictive features for hate speech detection on twitter, in: Proceedings of the NAACL student research workshop, 2016, pp. 88–93.
- [6] N. Jose, B. R. Chakravarthi, S. Suryawanshi, E. Sherly, J. P. McCrae, A survey of current datasets for code-switching research, in: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020.
- [7] R. Priyadharshini, B. R. Chakravarthi, M. Vegupatti, J. P. McCrae, Named entity recognition for code-mixed Indian corpus using meta embedding, in: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020.
- [8] B. R. Chakravarthi, Leveraging orthographic information to improve machine translation of under-resourced languages, Ph.D. thesis, NUI Galway, 2020.
- [9] B. R. Chakravarthi, N. Jose, S. Suryawanshi, E. Sherly, J. P. McCrae, A sentiment analysis dataset for code-mixed Malayalam-English, in: Proceedings of the 1st Joint Workshop on Spoken Lan-

- guage Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL), European Language Resources association, Marseille, France, 2020, pp. 177–184. URL: <https://www.aclweb.org/anthology/2020.sltu-1.25>.
- [10] B. R. Chakravarthi, M. Arcan, J. P. McCrae, Improving Wordnets for Under-Resourced Languages Using Machine Translation, in: Proceedings of the 9th Global WordNet Conference, The Global WordNet Conference 2018 Committee, 2018. URL: http://compling.hss.ntu.edu.sg/events/2018-gwc/pdfs/GWC2018_paper_16.
- [11] B. R. Chakravarthi, M. Arcan, J. P. McCrae, Comparison of Different Orthographies for Machine Translation of Under-Resourced Dravidian Languages, in: M. Eskevich, G. de Melo, C. Fäth, J. P. McCrae, P. Buitelaar, C. Chiarcos, B. Klimek, M. Dojchinovski (Eds.), 2nd Conference on Language, Data and Knowledge (LDK 2019), volume 70 of *OpenAccess Series in Informatics (OASICS)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2019, pp. 6:1–6:14. URL: <http://drops.dagstuhl.de/opus/volltexte/2019/10370>. doi:10.4230/OASICS.LDK.2019.6.
- [12] B. R. Chakravarthi, R. Priyadharshini, B. Stearns, A. Jayapal, S. S, M. Arcan, M. Zarrouk, J. P. McCrae, Multilingual multimodal machine translation for Dravidian languages utilizing phonetic transcription, in: Proceedings of the 2nd Workshop on Technologies for MT of Low Resource Languages, European Association for Machine Translation, Dublin, Ireland, 2019, pp. 56–63. URL: <https://www.aclweb.org/anthology/W19-6809>.
- [13] B. R. Chakravarthi, P. Rani, M. Arcan, J. P. McCrae, A survey of orthographic information in machine translation, arXiv e-prints (2020) arXiv:2008.
- [14] Y. Chen, Y. Zhou, S. Zhu, H. Xu, Detecting offensive language in social media to protect adolescent online safety, in: 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing, IEEE, 2012, pp. 71–80.
- [15] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, Predicting the type and target of offensive posts in social media, arXiv preprint arXiv:1902.09666 (2019).
- [16] G. K. Pitsilis, H. Ramampiaro, H. Langseth, Detecting offensive language in tweets using deep learning, arXiv preprint arXiv:1801.04433 (2018).
- [17] S. Kamble, A. Joshi, Hate speech detection from code-mixed Hindi-English tweets using deep learning models, arXiv preprint arXiv:1811.05145 (2018).
- [18] A. Mishra, S. Pal, IIT Varanasi at HASOC 2019: Hate speech and offensive content identification in Indo-European languages., in: FIRE (Working Notes), 2019, pp. 344–351.
- [19] B. R. Chakravarthi, M. A. Kumar, J. P. McCrae, P. B, S. KP, T. Mandl, Overview of the track on HASOC-Offensive Language Identification- DravidianCodeMix, in: Working Notes of the Forum for Information Retrieval Evaluation (FIRE 2020). CEUR Workshop Proceedings. In: CEUR-WS.org, Hyderabad, India, 2020.
- [20] B. R. Chakravarthi, V. Muralidaran, R. Priyadharshini, J. P. McCrae, 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), European Language Resources association, Marseille, France, 2020, pp. 202–210. URL: <https://www.aclweb.org/anthology/2020.sltu-1.28>.
- [21] B. R. Chakravarthi, R. Priyadharshini, V. Muralidaran, S. Suryawanshi, N. Jose, E. Sherly, J. P. McCrae, Overview of the track on Sentiment Analysis for Dravidian Languages in Code-Mixed Text, in: Working Notes of the Forum for Information Retrieval Evaluation (FIRE 2020). CEUR Workshop Proceedings. In: CEUR-WS.org, Hyderabad, India, 2020.
- [22] B. R. Chakravarthi, R. Priyadharshini, V. Muralidaran, S. Suryawanshi, N. Jose, E. Sherly, J. P. McCrae, Overview of the track on Sentiment Analysis for Dravidian Languages in Code-Mixed

- Text, in: Proceedings of the 12th Forum for Information Retrieval Evaluation, FIRE '20, 2020.
- [23] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).
- [24] F. Chollet, et al., keras, 2015.