Siva@HASOC-Dravidian-CodeMix-FIRE-2020: Multilingual Offensive Speech Detection in Code-mixed and Romanized Text

Siva Sai[a], Yashvardhan Sharma[b]

Birla Institute of Technology & Science, Pilani Campus
Pilani-333031

Abstract

Detecting and eliminating offensive and hate speech in social media content is an important concern as hate and offensive speech can have serious consequences in society ranging from ill-education among youth to hate crimes. Offensive speech identification in countries like India poses several additional challenges due to the usage of code-mixed and romanized variants of multiple languages by the users in their posts on social media. HASOC-Dravidian-CodeMix - FIRE 2020 extended the task of offensive speech identification to Dravidian languages. In this paper, we describe our approach in HASOC Dravidian Code-mixed 2020, which topped two out of three tasks(F1-weighted scores - 0.95 and 0.90) and stood second in the third task lagging the top model only by 0.01 points((F1-weighted score - 0.77). We propose a novel and flexible approach of selective translation and transliteration to be able to reap better results out of fine-tuning and ensembling multilingual transformer networks like XLM-RoBERTa and mBERT. Further, we implemented pre-trained, fine-tuned and ensembled versions of XLM-RoBERTa for offensive speech classification. We open source our work to facilitate further experimentation.

Keywords

Offensive speech detection, selective translation and transliteration, XLM-RoBERTa, Transformer Neural Networks.

1. Introduction

Offensive speech is defined as speech that causes a person to feel upset, resentful, annoyed, or insulted. In recent years, social media such as Twitter, Facebook and Reddit have been increasingly used for the propagation of offensive speech and the organization of hate and offense-based activities. In a country like India with multiple native languages, users prefer to use their regional language in their social media interactions. It has also been identified that users tend to use roman characters for texting instead of the native script. This poses a severe challenge for the identification of offensive speech.

Research in offensive and hate speech identification is slowly picking up the pace mainly due to a variety of shared tasks being organized[1, 2, 3, 4, 5]. Until a few years ago, hate and offensive speech were identified manually which is now an impossible task due to the enormous amounts of data being generated daily on social media platforms. The need for scalable, automated methods of hate speech detection has attracted significant research from the domains of natural language processing and machine learning. A variety of techniques and tools like bag of words models, N-grams, dictionary-based approaches, word sense disambiguation techniques are developed and experimented with by researchers. Recent developments in multilingual text classification are led by Transformer

https://github.com/SivaAndMe/Multilingual-Offensive-Speech-Detection-in-Code-mixed-and-Romanized-Text

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

f20170779@pilani.bits-pilani.ac.in (S. Sai); yash@pilani.bits-pilani.ac.in (Y. Sharma)

architectures like mBERT[6] and XLM-RoBERTa[7]. Transformer networks are proved to be better than all of the traditional techniques mentioned above. An additional advantage of these architectures, particularly XLM-RoBERTa, is that it yields good results even with lower resource languages and this particular aspect is beneficial to Indian languages which do not have properly established datasets. In our work, we focused on using these architectures in multiple ways. However, there is a caveat in directly using the models on the romanized or code-mixed text: the transformer models are trained on languages in their native script, not in the romanized script in which users prefer to write online. We solve this problem by using a novel way to convert the romanized sentences into their native language while preserving their semantic meaning - selective translation ad transliteration.

Our important contributions are as follows 1) Proposed selective translation and transliteration for text conversion in romanized and code-mixed settings which can be extended to other romanized and code-mixed contexts in any language 2) Experimented and analyzed the effectiveness of finetuning and ensembling of XLM-RoBERTa models for offensive speech identification in code-mixed and romanized scripts.

The rest of our paper is organized as follows: In section 2, we discuss related work followed by Task and dataset description in section 3. Section 4 describes methodology and section 5 discusses about internal evaluation results and official results. Finally, we conclude the paper with section 5-providing some insights for future research in this field.

2. Related works

Two major shared tasks organized in offensive language identification are OffensEval 2019[3] and GermEval[5]. OffensEval, organized as a part of SemEval-2019, used Offensive Language Identification Dataset (OLID) which consists of 14,000 English tweets extracted from Twitter. OffensEval 2020[4] extended its precursor with additional data and additional languages. The task focused on identification and categorization of offensive content and target identification for offensive posts. GermEval shared task focused on coarse-grained and fine-grained classification of offensive speech using German tweets from Twitter. Other tasks related to offensive speech identification include HASOC-19[1] which dealt with hate speech and offensive content identification in Indo-European languages, TRAC-2018[2], which dealt with aggression identification in Bengali, Hindi and English. HatEval 2019 dealing with coarse and fine-grained hate speech identification against immigrants and women also falls into the category of offensive speech identification. While HASOC-19 and TRAC 2020 dealt with offensive speech identification in Indian languages of Bengali and Hindi, HASOC-Dravidian-CodeMix - FIRE 2020 is the first shared task to conduct offensive speech identification task in Dravidian languages. This task also stands out from other tasks in using YouTube comments data, unlike other tasks which mostly focused on Twitter and Facebook posts.

Researchers used a wide variety of techniques for the identification of offensive language. Saha et al.[8], used LGBM classifier on top of the combination of multilingual BERT and LASER pre-trained embeddings in HASOC-19. Subhanshu et al.[9] fine-tuned monolingual and multilingual BERT based network models to achieve good results. An interesting aspect of their system is training over joint-labels which helps in sharing information between subtasks and addresses the data sparsity issues. Risch et al.[10] used an ensemble of BERT models with different random seeds, which is the inspiration behind our ensembling strategy with XLM-RoBERTa models. There has been less research on text classification in Dravidian languages and there is no research in offensive speech identification for Dravidian languages so far. Thomas et al.[11] uses a simple LSTM model for sentiment analysis in the Malayalam language. Our work adresses this gap of less research in offensive speech identification

methods for Dravidian languages and the systems proposed can be extended to other Indian and foreign languages as well.

3. Task Description

As a part of HASOC-Dravidian-CodeMix - FIRE 2020[12, 13, 14, 15, 16] there are two binary classification tasks with the second task having two subtasks. The objective of all of the tasks is same: given a Youtube comment, classify it as offensive or not offensive. But the format and language of data provided to different tasks are different: Code-mixed Malayalam for Task-1, Tanglish for Task-2a(henceforth referred as Tanglish) and Manglish for Task-2b(henceforth referred as Manglish). We have participated in all the tasks and made three submissions for each task using different neural network architectures.

4. System Description

4.1. Preprocessing

The organizers anonymized the comments data provided to participants and basic text cleaning like removal of URLs is performed beforehand. Further, we have done following pre-processing on text for all the tasks: a) Lower case Tanglish and Manglish words. This step is not performed for words written in Malayalam script in Task-1 as there is no such casing used in Malayalam script. b) Remove emojis from comments. c) Remove all special characters, numbers and punctuation. d) Remove user mentions as they generally do not carry any semantic meaning.

4.2. Selective translation and transliteration

This is a novel idea we have used to get a proper representation of text in the native script for the final neural architecture training. The pseudo-code for the proposed algorithm is given in Algorithm 1. The primary need for this step is as follows: Recent advancements in state-of-the-art multilingual NLP tasks are leaded by Transformer architectures like mBERT and XLM-RoBERTa which are trained on multiple languages in the native script but not in the romanized script. Hence to reap better results by fine-tuning these architectures, the text is to be in a native script(for example, Tamil text in Tamil script).

To convert text into the native script, we cannot rely on neural translation systems, particularly in tweets where users tend to write informally using multiple languages. Also, translating romanized non-English language words into that particular language does not make any sense in our context. For example, translating the word "Maram"(which means tree in Tamil) directly into Tamil would seriously affect the entire sentence's semantics. In many cases, proper translations from English to a non-English language would not be available for words. So, as a solution to this problem, we propose selective transliteration and translation of the text. In effect, this process of conversion of romanized text(for example, Tanglish) is to transliterate the native language(Tamil) words in the text into Tamil and translate the English words in the text into Tamil selectively. This separation of English words from native language words is done using a big corpus of English words from nltk-corpus. The idea of this selective conversion is based on the observation that in romanized native language comments(like Tanglish), users tend to use English words only when they can convey the meaning better with the English word or when the corresponding native language word is not much used in regular conversations. For example, the word "movie" is more preferred by Tamil-users than its

corresponding Tamil word. Furthermore, architectures like XLM-RoBERTa, which are trained on multilingual datasets, will be able to extract proper embeddings in this case.

The translation of words(not the complete sentences as described in the algorithm) is done using Google Translate API[1], and transliteration is done with the help of BrahmiNet API[2]. The detection of language script is carried out with the help of langdetect API[3]. We followed this step for all of the tasks. It is to be noted that in Tanglish and Manglish tasks, there are no words in the native script(Tamil or Malayalam).

Algorithm 1: Algorithm for Selective Translation and Transliteration of mixed-script romanized languages

Input : Preprocessed romanized or Code-mixed text T and desired native language for the final script L
Output: Text in native script

```
Initialization: EngWords = Set of all english words
words = splitSentIntoWords(T)
LOOP Process
for i=0 to len(words) do
    word = words[i]
    if(detectLanguageScript(word)==L) then
        continue
    else if(word in EngWords) then
        words[i] = translate(word,L)
    else
        words[i] = transliterate(word,L)
    endif
end for
return joinWordsToSent(words)
```

4.3. Models

Recent studies show that pre-trained word embeddings and fine-tuning of state-of-the-art Transformer architectures show better performance in text classification compared to classical machine learning approaches like N-gram features with bag of words models(which include count vectorizer or TF-IDF features). So we directed our entire focus on using the word-embeddings of transformer architectures both pre-trained and fine-tuned for text classification. The text obtained using selective translation and transliteration is used in further steps.

4.3.1. Pre-trained embeddings

Transfer learning using pre-trained word embeddings is proved to be useful for text classification in past offensive speech detection tasks[8]. So we experimented with XLM-RoBERTa pre-trained embeddings in our work.

[1]https://pypi.org/project/googletrans/
[2]http://www.cfilt.iitb.ac.in/brahminet/static/rest.html
[3]https://pypi.org/project/langdetect/

XLM-R embeddings XLM-RoBERTa is a large multilingual model trained on 2.5TB of CommonCrawl data in 100 different languages. It shows improved performance on low-resource languages and outperforms other transformer models like mBERT on cross-lingual benchmarks. The pre-trained model takes text as input and outputs feature vector of size 1024 for each token in the sentence. We take the average of the feature vectors for all tokens as the final feature vector for the entire sentence.

The pretrained feature vectors of size 1024 are given as input to classical classification algorithms like Logistic Regression. We performed an exhaustive classifier search among 16 classifiers like DecisionTreeClassifier, XGBoostClassifier, etc., to find the classifier that performs better for each task. Our observations show that Logistic Regression outperforms others for Tanglish and Manglish tasks, whereas MLP classifier shows better performance for Task-1. We also experimented with a neural network classifier on top of the word embeddings in place of classical algorithms. However, it did not improve performance much. We have used the Pytorch framework to obtain pre-trained embeddings and Sklearn for classifiers.

4.3.2. Fine-tuning Transformer architectures

When we extract features from the pre-trained model, we are using the base model as it is. However, we can fine-tune the base model to customize on our dataset to improve performance. We used Multilingual BERT(uncased), XLM-RoBERTa(both base and large versions) for fine-tuning. We performed minimal hyperparameter tuning. Early stopping with a patience of 10 is also used targeting the f1-weighted score, the final evaluation metric specified by the organizers. A maximum sequence tokens length of 70 is used for all the models based on the observation that around 95% of comments have number of tokens less than 70. We have evaluated the model once for every 100 batches during fine-tuning with 50 as maximum number of epochs. It implies that the model is evaluated once for every 1.25 epochs.

4.3.3. Ensembling Transformer architectures

The instability and variance of the transformer architectures' performance is the motivation behind this ensembling strategy[10]. Devlin et al.[6] show that the performance(accuracy score) of the BERT model on small datasets, such as the Microsoft Research Paraphrase Corpus (MRPC), varies between 84% and 88%. In our experiments with XLM-R base models, we observed a similar pattern:± 5% F1-weighted score for Task-1 and Manglish, and ±4% F1-weighted score for Tanglish on validation data. This variance can be created by slight changes in hyperparameters(random seed particularly) and training data. The random seed of a model affects the initialization of weights of the final classification layer. Furthermore, change in random seed while splitting the data into train and validation sets decide which samples go into each of them. This also affects the ordering of the samples in a particular set.

We experimented with 10 different random seeds on XLM-R base model. Risch et al.[10] reports that 15 is the optimal number of BERT models for ensembling and that the performance of plateaus above that number. In our experiments with XLM-R base, we observed that ensembling 10 models is optimal, where each model corresponds to a different random seed. All other hyper-parameters are kept the same for ensembling. As we are using Bagging ensemble strategy, we used soft-majority voting to combine predictions of these ten models. Soft majority voting simply adds the probabilities of each class from all the models and chooses the one with highest probability as the predicted class.

Apart from ensembling of same XLM-RoBERTa models, we also experimented with ensembles different models : XLM-RoBERTa base + XLM-RoBERTa large and XLM-RoBERTa base + mBERT to

Table 1
Internal evaluation results - F1-Weighted

Task	XLMR-pretrained+clf	mBERT	XLMR-B	XLMR-B+ mBERT	XLMR-B + XLMR-L
Task-1	0.88	0.91	**0.93**	**0.93**	0.93
Tanglish	0.78	0.83	0.86	**0.88**	0.85
Manglish	0.67	0.60	**0.69**	**0.69**	**0.69**

Table 2
Official results - F1-Weighted

Task	Task-1	Tanglish	Manglish
Our team	**0.95**	**0.90**	0.77
Other best performing team	0.95	0.88	**0.78**

analyse the effect of diversity of architectures on the performance.

5. Results and Discussion

To test the performance of our proposed models, we had evaluated them on validation data before the test set was made available by the organizers. When unlabeled test data was released, we used the above-mentioned validation data as dev set to develop the model, and final predictions are made using these models. Ensemble of XLM-RoBERTa models is not experimented during internal evaluation due to computational limitations. Nevertheless, we verified the better performance of Ensemble models with few random seeds and directly used them for final training.

5.1. Internal evaluation results on validation data

As discussed previously, we created a stratified dev set from training data for Tanglish and Manglish. The dev set for Task-1 is provided by organizers. And these dev sets are used for final internal valuation as test sets. A 10% of training data was used as dev data for training for all the tasks. The results are shown in Table 1. In the case of pre-trained embeddings, clf(classifier) used is Logistic Regression for Tanglish and Manglish tasks and MLP for Task-1. Because XLM-RoBERTa is also trained on Tamil dataset in roman script, we experimented with directly feeding the pre-processed Tanglish text to the model without selective conversion. But, the performance is significantly lesser than the model which follows entire pipeline. It can be observed from the table that fine-tuning models gave better results than directly using off-the-shelf embeddings. But for Manglish, where the F1-weighted score is relatively less than other tasks, pre-trained embeddings come closer to fine-tuned models in performance. The superior performance of XLMR-base model in all the tasks can also be inferred from the table. This is the reason, we chose XLMR-base for ensembling strategy.

5.2. Submission results

Participants are not provided with gold labels for the test data, and the final evaluation is done by the organizers themselves. So in this section, we can provide only the results given by organizers. Also, organizers provided the scores only for the best performing system among three submissions

for each task. The official results of our team are shown in Table 2. Our team - Siva(SivaSai@BITS on leaderboard) secured first position in Task-1 and Tanglish and the second position in Manglish. Also, we can see that our official results are slightly better than internal evaluation results for Task-1 and Tanglish task. And the official results are significantly higher than internal evaluation results for Manglish task(8% higher). Although it is not clear which submission of our team for each task gave these results as the organizers did not mention run-numbers in final results, we firmly believe that XLMR-ensemble is the winning model in all the tasks. It can be inferred from both internal evaluation results and official results that performance scores for Manglish are significantly less than those of Tanglish task(10% lesser). We attribute this to agglutinative and inflectional nature of Malayalam language.[4]

6. Conclusion and Future work

In this paper, we have presented details about our submission in HASOC-Dravidian-CodeMix - FIRE 2020. A novel technique of selective translation and transliteration is proposed to deal with code-mixed and romanized offensive speech classification in Dravidian languages. This technique is flexible and can be extended to other languages and other classification problems. For classification, classical classifiers on top of pre-trained embeddings, fine-tuned XLM-RoBERTa models, and an ensemble of XLM-RoBERTa models are used. Our work also points the usefulness of Transformer architectures, particularly XLM-RoBERTa, for low resource languages like Tamil and Malayalam.

As observed in section 5.2, the relatively bad performance of classifiers in Manglish task compared to Tanglish task can be attributed to the agglutinative and inflectional nature of the Malayalam language. Hence, future work can focus on developing techniques and tools to deal with this characteristic of Malayalam language. In this work, we have not experimented with ensemble models of classical classifiers like logistic regression on top of word embeddings and transformer networks, which also poses another direction to future work in the field. Further, researchers can also focus on analyzing the use of pre-trained mBERT embeddings and emoji to text conversion for better feature representation.

7. Acknowledgement

The authors would like to convey their sincere thanks to the Department of Science and Technology (ICPS Division), New Delhi, India, for providing financial assistance under the Data Science (DS) Research of Interdisciplinary Cyber Physical Systems (ICPS) Programme [DST/ICPS/CLUSTER/Data Science/2018/Proposal-16:(T-856)] at the department of computer science, Birla Institute of Technology and Science, Pilani, India.

References

[1] T. Mandl, S. Modha, P. Majumder, D. Patel, M. Dave, C. Mandlia, A. Patel, Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages, in: Proceedings of the 11th Forum for Information Retrieval Evaluation, 2019, pp. 14–17.

[4] https://thottingal.in/blog/2017/11/26/towards-a-malayalam-morphology-analyser/

[2] R. Kumar, A. K. Ojha, S. Malmasi, M. Zampieri, Benchmarking aggression identification in social media, in: Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), 2018, pp. 1–11.
[3] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval), arXiv preprint arXiv:1903.08983 (2019).
[4] M. Zampieri, P. Nakov, S. Rosenthal, P. Atanasova, G. Karadzhov, H. Mubarak, L. Derczynski, Z. Pitenis, Ç. Çöltekin, Semeval-2020 task 12: Multilingual offensive language identification in social media (Offenseval 2020), arXiv preprint arXiv:2006.07235 (2020).
[5] J. M. Struß, M. Siegel, J. Ruppenhofer, M. Wiegand, M. Klenner, et al., Overview of germeval task 2, 2019 shared task on the identification of offensive language (2019).
[6] 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).
[7] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov, Unsupervised cross-lingual representation learning at scale, arXiv preprint arXiv:1911.02116 (2019).
[8] P. Saha, B. Mathew, P. Goyal, A. Mukherjee, Hatemonitors: Language agnostic abuse detection in social media, arXiv preprint arXiv:1909.12642 (2019).
[9] S. Mishra, S. Mishra, 3idiots at hasoc 2019: Fine-tuning transformer neural networks for hate speech identification in Indo-European languages., in: FIRE (Working Notes), 2019, pp. 208–213.
[10] J. Risch, R. Krestel, Bagging bert models for robust aggression identification, in: Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying, 2020, pp. 55–61.
[11] M. Thomas, C. Latha, Sentimental analysis of transliterated text in malayalam using recurrent neural networks, Journal of Ambient Intelligence and Humanized Computing (2020) 1–8.
[12] 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.
[13] 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: Proceedings of the 12th Forum for Information Retrieval Evaluation, FIRE '20, 2020.
[14] 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.
[15] 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 Language 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.
[16] B. R. Chakravarthi, Leveraging orthographic information to improve machine translation of under-resourced languages, Ph.D. thesis, NUI Galway, 2020. URL: http://hdl.handle.net/10379/16100.