Offensive text prediction using Machine Learning and Deep Learning approaches

Bhuvana Jayaraman[1], Mirnalinee T.T[1], Karthik Raja Anandan[1], Aarthi Suresh Kumar[1] and Anirudh Anand[1]

[1]*Department of CSE*
Sri Sivasubramaniya Nadar College of Engineering,
Chennai, Tamil Nadu, India

Abstract

As communities become multilingual, code-mixing becomes a prevalent phenomenon. Traditional models trained on monolingual data have proved to be inadequate for code-mixed data. This work was submitted for the Dravidian-CodeMix-HASOC 2021 task on offensive language detection.The paper depicts a deep learning and Naive Bayes based approach towards solving the problem of understanding text written in non-native script and classifying them into offensive and not-offensive text. A Bidirectional Long Short Term Memory (BiLSTM) model is employed to solve this classification task. The model is tested for F1 score to obtain the degree to which the model correctly classifies the text as offensive or non-offensive, where the BiLSTM is observed to have performed well than the Naive Bayes approach

Keywords

Bidirectional LSTM, Code-mixed, Offensive language identification, Sentiment analysis, Tokenizer, Naive Bayes, Tamil, Malayalam, Dravidian Languages

1. Introduction

Recently, social media has become the most sought after mode of practicing one's right to speech. The inadequacy of the rules imposed by social media platforms has led to the rise of offensive speech that tends to destabilize communities. Additionally, most of the social media comments are made in non-native script. This code mixed text makes the task even more challenging as models trained on monolingual data fail on code-mixed data due to the complexity of code-switching at different linguistic levels in the text. It is difficult to transliterate Tamil to English effectively using the libraries *indic-nlp*. Most of the Tamil text did not match with its English equivalent because of the letters that are unique to Tamil and do not exist in the English scripture. Additionally, Tamil has many words that have different meanings in different contexts and predicting the offensiveness of the comment was a challenging task.

In the current scenario people are working on newer models like BERT and their variants are

Forum for Information Retrieval Evaluation, December 13-17, 2021, India
bhuvanaj@ssn.edu.in (B. Jayaraman); mirnalineett@ssn.edu.in (M. T.T); karthikraja19048@cse.ssn.edu.in (K. R. Anandan); aarthi19003@cse.ssn.edu.in (A. S. Kumar); anirudh19015@cse.ssn.edu.in (A. Anand)
https://www.ssn.edu.in/staff-members/dr-j-bhuvana/ (B. Jayaraman);
https://www.ssn.edu.in/staff-members/dr-t-t-mirnalinee/ (M. T.T)
0000-0002-9328-6989 (B. Jayaraman); 0000-0001-6403-3520 (M. T.T)

Language	Training	Validation
Tamil	3999	939
Malayalam	3999	950

Table 1
Number of training and validation samples

currently employed and researched upon for offensive language classification. Here we have used BiLSTM for detecting offensive tweets.

2. Literature Survey

Anti social comments in YouTube have been classified into offensive and non offensive in [1]. Initially the input is tokenized, filtered and normalized as the part of pre-processing steps. Supervised machine learning technique namely, Support Vector Machine (SVM) has been applied with the help of word level features and achieved F1 score of 82%. Authors in [2] studied the performance of the dataset fused with augmented data to detect the offensive content. The original data is 100k tweets of ICWSM-2020 Data Challenge Task. To balance the class imbalance problem, generation-based data augmentation techniques named Dager was employed. The work shown an 22% increase in F1 score with fusing ratio of Dager generated data and original data to a ratio of 70% and 10% of original data.

Tweets are preprocessed to remove punctuation, repeated characters, non Arabic characters before classifying them into offensive or not. Dataset belonged to OSACT Shared task having 10,000 Arabic tweets [3]. Several classifiers were employed in this work, including the combination architecture of CNN-BiLSTM, M-BERT and SVM. Among which the SVM outperformed the other classifiers with 97.1% accuracy.

KanCMD dataset with 7,671 post has code mixed text in Kannada posted as YouTube comments [4]. Two tasks that have performed are Sentiment Analysis and offensive Language classification with six classes. A variety of machine learning techniques have been used to perform both the tasks. Authors observed that almost all the algorithms employed have performed poorly, out of which Random Forest and logistic regression have outperformed others with around 66% accuracy for offensive Language classification.

Four different models have been used to detect offensive language [5]. SVM, Naive Bayes have used TF-IDF vector as features, LSTM with attention is fed with GloVe word embeddings applied on 220,000 comments,taken from toxic comments dataset with six classes. LSTM has provided 99.67% of Explanatory Power Index measure. Similar researches on offensive language identification are found in [6], [7], [8] and [9].

3. Datasets

The given datasets are already split into 2 files for training and validation separately. The training file for Tamil language comprises of 1980/3999 offensive labels and validation set

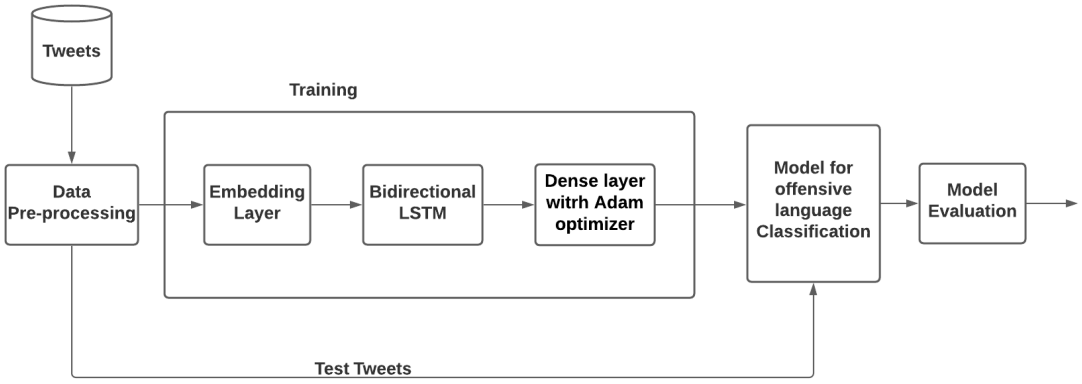

Figure 1: Flowchart of the proposed BiLSTM Model

contains 474/939 offensive labels as shown in Table 1. Similarly Malayalam dataset also contains 2047/3999 and 478/950 offensive labels in corresponding training and validation sets [9]. As we can see the two classes are almost equally balanced and hence do not require any kind of further preprocessing to prevent over fitting of the dominant class.

4. Methodology

The proposed architecture uses both a machine learning classifier and a deep learning architecture for offensive tweet classification. BiLSTM [10] and Naive Bayes classifier [11] are the two chosen approaches for the given shared task.

4.0.1. Deep Learning Model Architecture: BiLSTM

Our method employs a bidirectional LSTM to classify comments as offensive and not offensive. A bidirectional LSTM runs two LSTMs to read the text from front and back simultaneously. This marks the basis of difference between LSTM and BiLSTM. LSTM stores only the past while BiLSTM stores both the past and future. This makes a Bi-LSTM more efficient in understanding the mood and context of the comment when compared to other models.

$$\overrightarrow{h} = f(\overrightarrow{W}x_t + \overrightarrow{V}.\overrightarrow{h}_{t-1} + \overrightarrow{b}) \tag{1}$$

$$\overleftarrow{h}_t = f(\overleftarrow{W}x_t + \overleftarrow{V}.\overleftarrow{h}_{t+1} + \overleftarrow{b}) \tag{2}$$

$$\hat{y}_t = g(Uh_t + c) = g(U[\overrightarrow{h_t}; \overleftarrow{h_t}] + c) \tag{3}$$

The Equations 1, 2, 3 [12] render the base for computation at a layer in a bidirectional lstm model. The score results from the forward LSTM and backward LSTM are concatenated as given in the third equation. Mathematically, at time *t*, an intermediate neuron receives one set of parameters from the previous time-step of the same LSTM layer and two sets of parameters from the previous LSTM layer, one from the forward and the other from the backward LSTM. The

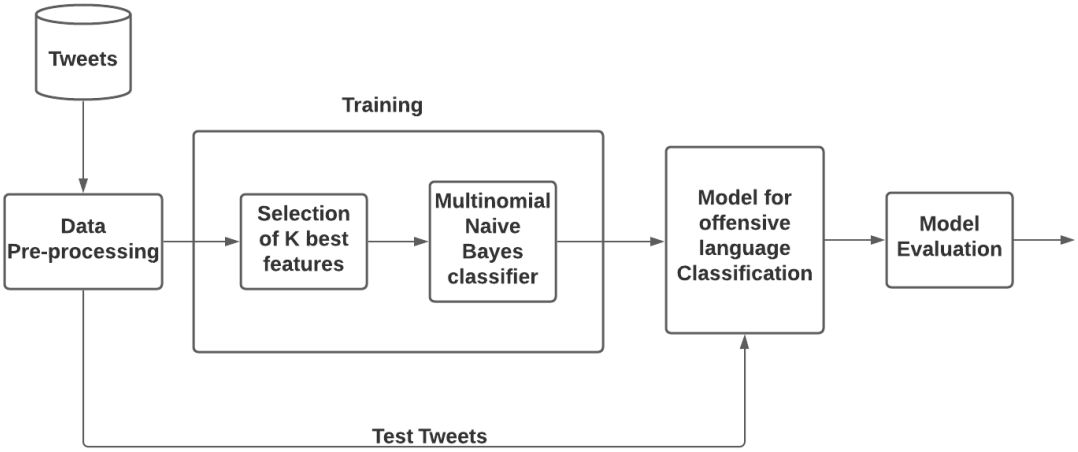

Figure 2: Proposed Naive-Bayes Model

aggregated sum of results of all the hidden states is passed to the softmax activation function to compute the probability of each class.

The first layer of the model constitutes the embedding layer that converts the tokenized words into fixed-sized vectors. The Embeddings layer in Keras module was used to obtain a dense vector representation for words which learns the relational order of words and this adjusted embedding is fed to the BiLSTM layer [13].

This BiLSTM layer learns the context of sentences and learns the weightage of every word or a group of words in deciding whether it is offensive or not. BiLSTM also considers the order of occurrence of the words which makes the model more efficient in determining the context and mood of the sentence. The proposed architecture with BiLSTM is shown in Figure 1. The loss function used in Binary cross entropy which is optimized using the Adam optimizer with a learning rate of 0.01. The model performance during training is evaluated using the accuracy metric.

4.1. Naive Bayes Model

The Naive Bayes model uses an approximation of the well known Bayes theorem. It takes the probability of each feature given an output to predict the probability of classifying the given input to a particular output. This algorithm is widely used in sentimental analysis with large datasets. The model architecture is shown in Figure 2. Bayes Theorem and Naive Bayes Theorem are represented as in Equation 4, 5.

$$P(Y|x_1x_2 * ... * x_n) = P(Y)\frac{P(x_1x_2...x_n|Y)}{P(x_1x_2...x_n)} \tag{4}$$

$$P(Y|x_1x_2...x_n) = P(Y)\frac{P(x_1|Y)P(x_2|Y)...P(x_n|Y)}{P(x_1x_2...x_n|Y)} \tag{5}$$

As we can see from above equations, Naive Bayes theorem assumes conditional independence of each feature present in the input. Therefore it fails to account for cases where a set of words determine the offensiveness of a tweet rather than individually.

5. Implementation and Experiments

5.1. LSTM Pre-processing

The redundant rows containing no meaningful tweets were removed as the first step of preprocessing the training dataset for BiLSTM network. The tweets were converted to lower case to avoid the same words being counted as separate words. Then the tweets were tokenized using the Keras Tokenizer, which splits the sentences into words and assigns a unique number to each word. The text to sequences function fits the encoding on the training data converting the string of words to that of numbers. The tokenized tweets were pre-padded to a uniform length of 100 tokens. Since, the number of words is not constant in the dataset, a safe upper bound of 100 words was chosen to fit the longest sentences in the dataset as well. Finally, the column containing the output was converted into categorical data. Since the training dataset was limited to non-native script and script without emoticons, no special libraries were used for transliteration or demotifying the text.

The same pre-processing steps as the training dataset are applied to the testing dataset. Additionally, the testing dataset contained comments in native script and raw data with emoticons. To deal with such data, the following libraries were used:

- `langdetect` : To detect the language of the script based on individual characters in the text
- `indic-transliteration` : To transliterate native script to roman script
- `re` : To identify and remove emoticons from the comments

A single Embedding layer is used, whose output is fed to a BiLSTM layer with an output dimension of 150. The output layer has an activation function of softmax. Adam optimizer with a learning rate of 0.01 and binary_crossentropy loss function were used to train the model.

5.1.1. Pre-processing for Naive Bayes

The train set is initially cleaned off punctuation and other English stop words with the help of stop word and punctuation corpus in *nltk*.The cleaned data is sent to another layer for transforming emojis found in the tweets to their corresponding emotions.Additionally a English language based Wordnet lemmatizer was used which didn't add up to any further improvement (add this to modifications) , The transformed tweets are used to generate a bag of words count vector. The mutual info classify function was used to obtain the k strongest word features from the dataset. k is iteratively set to the numbers in the set { 100, 500, 1000} This is used to train the multinomial naive Bayes model.The test data and validation data follow the same preprocessing procedure.

Multinomial Naive Bayes was used directly from the *sklearn* library with the default parameters with Additive Laplace smoothing parameter as 1.

Model	Language	Training Accuracy	Validation Accuracy	F1-score
BiLSTM	Tamil	0.8170	0.8296	0.6628
BiLSTM	Malayalam	0.5414	0.5558	0.6979
NaiveBayes	Tamil	0.8487	0.839	0.804
NaiveBayes	Malayalam	0.7432	0.6804	

Table 2
Performance of proposed Models during training

Computation is done in Python using Google Colab notebook and its GPU was used to train the model [1]. A general purpose RAM size of 8GB was allotted with a 2.3GHz Intel Xenon CPU.

6. Results

TeamName	Precision	Recall	F1-Score	Rank
AI_ML_NIT_Patna	0.539	0.509	0.515	15
JBTTM	0.537	0.483	0.503	16

Table 3
Tamil codemix Task

TeamName	Precision	Recall	F1-Score	Rank
MUM	0.628	0.637	0.632	10
JBTTM	0.577	0.584	0.58	11

Table 4
Malayalam codemix Task

Table 2 shows the performance of the two models on two languages Tamil and Malayalam with respect to accuracy. The validation accuracy of Naive Bayes on both languages is observed to be greater than the validation accuracy of BiLSTM model. Naive Bayes model could only give a maximum validation accuracy of 68% even after changing the number of best features chosen from input at training. But with respect to the languages the models trained for Tamil language fared well than the model for offensive language prediction for Malayalam and this can be attributed to the class imbalance of samples in the dataset.

[1] https://github.com/ask-1710/FIRE2021-OffensiveLanguageDetection.git

The two proposed models are submitted for evaluation in HASOC-Dravidian CodeMix Shared Task on Offensive Language Detection in Tamil and Malayalam. Out of which the deep learning BiLSTM model was evaluated to be placed in rank 16 and rank 11 in Tamil Task and Malayalam task respectively based on the recall, precision and F1-score as shown in Tables 3 and 4. The BiLSTM model which performed well on Tamil language during training did not reflect the same when it was tested on unseen data. It has achieved only 55% accuracy in validation for Malayalam language but performed better than its Tamil counterpart in testing.

The Naive Bayes model was not ranked may be due to the conditional independence assumed by that approach on each feature of the input, and not fare well. Therefore it would have failed to account for cases where a set of words determine the offensiveness of a tweet rather than individually.

7. Conclusion

To predict the offensiveness of the text, one machine learning based Naive Bayes approach and a deep learning based BiLSTM were proposed in this work. The deep learning based model has not performed well in training when compared with Naive Bayes approach but have shown a noticeable performance during testing and placed in 16 and 11 ranks for Tamil and Malayalam languages respectively. Other classic algorithm Naive Bayes were also tested and not considered on the basis of accuracy values. From the results obtained it can be observed that that our proposed BiLSTM model too gives lower accuracy on unseen data. In the future we plan to use an ensemble model consisting of BiLSTM and BERT models to get better results. The performance can also be improved by employing augmentation techniques to handle the class imbalance problem.

References

[1] A. Alakrot, L. Murray, N. S. Nikolov, Towards accurate detection of offensive language in online communication in Arabic, Procedia computer science 142 (2018) 315–320.

[2] R. Liu, G. Xu, S. Vosoughi, Enhanced offensive language detection through data augmentation, arXiv preprint arXiv:2012.02954 (2020).

[3] H. Mubarak, K. Darwish, W. Magdy, T. Elsayed, H. Al-Khalifa, Overview of OSACT4 Arabic offensive language detection shared task, in: Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, 2020, pp. 48–52.

[4] A. Hande, R. Priyadharshini, B. R. Chakravarthi, KanCMD: Kannada CodeMixed dataset for sentiment analysis and offensive language detection, in: Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media, 2020, pp. 54–63.

[5] J. Risch, R. Ruff, R. Krestel, Offensive language detection explained, in: Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying, 2020, pp. 137–143.

[6] S. Saumya, A. Kumar, J. P. Singh, Offensive language identification in Dravidian code

mixed social media tex, in: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, 2021, pp. 36–45.
[7] K. Yasaswini, K. Puranik, A. Hande, R. Priyadharshini, S. Thavareesan, B. R. Chakravarthi, IIITT@ DravidianLangTech-EACL2021: Transfer Learning for Offensive Language Detection in Dravidian Languages, in: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, 2021, pp. 187–194.
[8] D. Sivalingam, S. Thavareesan, OffTamil@ DravideanLangTech-EASL2021: Offensive Language Identification in Tamil Text, in: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, 2021, pp. 346–351.
[9] B. R. Chakravarthi, P. K. Kumaresan, R. Sakuntharaj, A. K. Madasamy, S. Thavareesan, P. B, S. Chinnaudayar Navaneethakrishnan, J. P. McCrae, T. Mandl, Overview of the HASOC-DravidianCodeMix Shared Task on Offensive Language Detection in Tamil and Malayalam, in: Working Notes of FIRE 2021 - Forum for Information Retrieval Evaluation, CEUR, 2021.
[10] L. Deng, Y. Liu, Deep learning in natural language processing, Springer, 2018.
[11] I. Rish, et al., An empirical study of the naive Bayes classifier, in: IJCAI 2001 workshop on empirical methods in artificial intelligence, volume 3, 2001, pp. 41–46.
[12] R. Socher, R. S. Mundra, CS 224D: Deep Learning for NLP1 (2016).
[13] J. Xie, B. Chen, X. Gu, F. Liang, X. Xu, Self-attention-based BiLSTM model for short text fine-grained sentiment classification, IEEE Access 7 (2019) 180558–180570.