HRS-TECHIE@Dravidian-CodeMix and HASOC-FIRE2020: Sentiment Analysis and Hate Speech Identification using Machine Learning, Deep Learning and Ensemble Models

Sridhar Swaminathan[a], Hari Krishnan Ganesan[b], and Radhakrishnan Pandiyarajan[c]

[a] *Department of Computer Science Engineering, Bennett University, Greater Noida 201310, India*
[b] *Department of Computer Science & Engineering, University College of Engineering, Trichy 620024, India*
[c] *Department of Information Technology, University College of Engineering, Trichy 620024, India*

Abstract

In this paper, we (HRS-TECHIE) present our submissions to challenges Dravidian-CodeMix and HASOC at FIRE 2020. Classification of sentiments from social media posts and comments is essential in this modern digital era. Dravidian-CodeMix (Sentiment analysis for Dravidian Languages in Code-Mixed Text) at FIRE 2020 is a challenge for classification of sentiments of YouTube comments posted in mix of Tamil-English (Task 1) and Malayalam-English (Task 2) languages. Our chosen task is to classify YouTube comments written in Tamil-English into one of five types of sentiment classes. Identification of hate speech, offensive and profane contents from social media posts and comments is essential in this modern digital era for preventing individuals in the digital media from the cyber harassment. HASOC 2020 (Hate Speech and Offensive Content Identification in Multiple Languages) at FIRE 2020 is a challenge of identifying the bullying content from Twitter comments posted in English, German and Hindi (subtask A) languages and further classifying the type of bullying present in that comment for each language (subtask B). We worked on both subtasks A and B for the English language to identify the bullying comment and type of bullying from Twitter comments. As part of these two challenges, we submitted different state-of-the-art machine learning and deep learning models for text classification. The models trained for sentiment classification task in Dravidian-CodeMix are Naïve Bayes, Decision tree, Random Forest, AdaBoost and Long Short Term Memory (LSTM). The models trained for hate speech and offensive content identification are Naïve Bayes, SVM, Decision tree, Random Forest, Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). We have also developed an ensemble of Machine Learning classifiers for both challenges. In Dravidian-CodeMix, we have achieved the best weighted F1-score 61% for both Naïve Bayes and LSTM models where weighted average F1-score of 60% was achieved for ensemble approach. In HASOC, we have achieved the best Macro average F1-score of 50.02% from LSTM model for subtask A and Macro average F1-score of 24.26% from ensemble approach for subtask B on the private test data.

Keywords

Sentiment Analysis, Cyber Bullying, TF-IDF, Word Embedding, Naïve Bayes, Decision tree, Random forest, AdaBoost, LSTM, GRU, Ensemble Learning.

1. Introduction

In this era of digital world, people are connected with each other via social media such as Twitter, YouTube, Instagram, etc. History of Natural Language Processing (NLP) dates back to 1950s however research on people's opinion or sentiments were not given much attention till 2005s [1].

FIRE '20, Forum for Information Retrieval Evaluation, December 16–20, 2020, Hyderabad, India
EMAIL: sridhar.swaminathan@bennett.edu.in (A. 1); harikrishnan12b2@gmail.com (A. 2); krishnan1998smart@gmail.com (A. 3)
* All authors contributed equally in this work
ORCID: 0000-0003-3446-9555 (A. 1); 0000-0002-1517-2708 (A. 2); 0000-0002-8074-4469 (A. 3)


Sentiment analysis uses the concept of the Natural Language Processing and Text Analysis to identify the type of the comment or review based on their emotions i.e. sentiments. Generally, sentiment analysis algorithms can be categorized into two broad categories such as knowledge based (language dependent) and statistics-based (language independent) approaches [2]. While there is a lot of research work for sentiment analysis for English language, only in the recent times researchers started working on regional languages of their countries. Recently many sentiment analyses approaches [3, 4] have been proposed for Indian languages also.

Dravidian-CodeMix challenge [5] at FIRE2020 [6] presents two sentiment analysis datasets based on Tamil-English [7] and Malayalam-English [8] languages. In this challenge, we focus on sentiment analysis of Tamil-English sentences (called as "Tanglish") written by social media users. Existing work [9,10] on the Tamil-English sentiment analysis are based on language specific as well as language independent approaches.

Social media has become one of the major ways of social interaction. In order to prevent users from the cyber bullying or cyber harassment (i.e. harassment using electronic medium), we must identify the comments in the social media whether it is harmful, hateful or offensive to any person or organization in the real life [11]. Hate speech and offensive content identification uses the concept of the Natural Language Processing and Text Analysis to identify the offensiveness of the comments [12]. Recently considerable amount of research work is going on to avoid Cyber Bullying or Hate Speech and Offensive Content Identification for English language [13,14].

HASOC 2020 challenge at FIRE2020 presents three Hate Speech and Offensive Content Identification [13] datasets based on English, German and Hindi languages [15,16]. In this challenge, we focused on Hate Speech and Offensive Content Identification of English sentences written by social media users [17,18]. In this paper, we present our submissions to the Dravidian-CodeMix and HASOC 2020 challenges at FIRE 2020 event.

2. Related Work

In this section, we will discuss about some related works for the sentiment analysis of Tamil language and hate speech detection.

2.1. Sentiment Analysis

Veena P V, et al.,[19] presented the techniques for language identification for code-mixed data i.e. Tamil-English, collected from Facebook. They used the word-embedding via character-embedding, then that word embedding features are converted into n-gram models like trigram and 5-gram. After this, SVM classifier is used to identify the language from the comments. Initially, they trained the SVM model by 10-fold cross validation technique to acquire the best model. They achieved 94% F1-score for both trigram and 5-gram model.

Rajat Singh, et al., [20] gives the idea of leverage the contextual property of the code-mixed data by giving them into Word2Vec model with skip-gram which used to extract the candidate words and their related variations using clustering. Additionally, these words are converted into unigram, bigram, TF-IDF vectors. Then these preprocessed texts will be fed into machine learning models like Naïve Bayes, SVM for the sentiment analysis and they achieved 63% and 53% F1-scores.

B. R. Chakravarthi, et al., [7] developed the state of art machine learning algorithms like Naïve Bayes, SVM, k-NN, Decision tree, Random forest, Logistic Regression with TF-IDF vectors,1D Conv-LSTM with embedding for the sentiment analysis of Tamil-English mixed data. They achieved the 65% F1-score for Random Forest classifier.

2.2. Hate Speech Detection

Thomas, et al., [12] used the logistic regression, naïve Bayes, decision trees, random forests, and linear SVM model for the automated hate speech detection. They tested each model using 5-fold cross validation method to prevent over fitting and select the best dataset. Also they used grid-search

method to repeat all of the models and parameters to find the best model with best parameter. Logistic Regression and Linear SVM performs considerably better than other models. The authors used the logistic regression with L2 regularization for the final model.

Marcos, et al., [14] used the OLID dataset for the offensive content identification problem. They designed the multi-level model in three levels. In first level SVM algorithm is used for the offensive content identification, then Bi-LSTM is used for categorizing the offensive content and at last, CNN is used for identifying the victim/target. They achieved the 80%, 66% and 47% of F1-macro score on each level respectively.

Bashar, et al., [16] also developed the multi-level model in three levels for the hate speech & offensive content classification. They used the Hindi language dataset where data collected from social media like Twitter. They used DNN, SVM, k-NN, Boosting, CNN for the offensive content identification of Hindi language. They achieved the F1-macro scores 80%, 56%, 51% for each level of identification, respectively.

3. Dataset Description

The Dravidian-CodeMix Dataset was collected from YouTube comments written in combination of Tamil and English languages (Task 1). Each comment belongs to one of 5 types of sentiment classes such as Positive, Negative, Mixed Feelings, Unknown State and Non-Tamil. "Thalivaaa…. nee vera level massss…" is a sample comment with positive sentiment written majorly in Tamil language with few English words. Number of comments in Training, Validation and Testing sets depicted in the table 1. It can be seen that majority of comments are having positive sentiment in the dataset. It should be noted that this class imbalance problem should be handled while training a text classification model.

Table 1
Distribution of YouTube comments in the training, validation, and testing dataset according to labels

Labels	Number of comments in data sets			Distribution of Classes
	Training set	Validation set	Testing set	
Mixed feelings	1283	141	377	11%
Negative	1448	165	424	13%
Positive	7627	857	2075	68%
Not-Tamil	368	29	100	03%
Unknown-state	609	68	173	05%
Total	11335	1260	3149	

Figure 1 shows the word cloud of top 100 most frequent words from the dataset. It can be seen that most of the frequent words are related to the positive or negative sentiments.

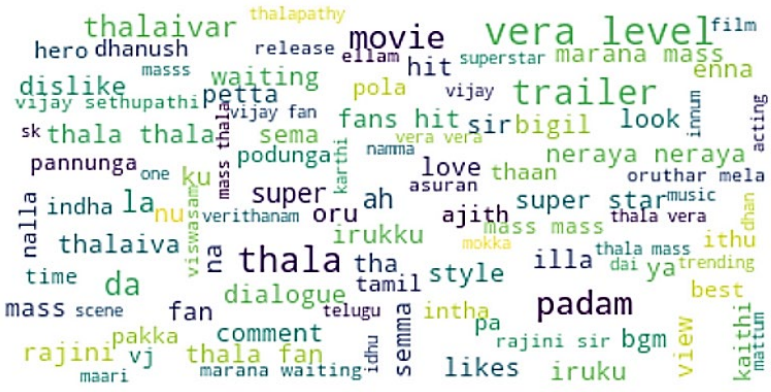

Figure 1: Word cloud of top 100 most frequent words the dataset

The HASOC 2020 Dataset was collected from Twitter comments written in English language. Each comment has two results with respect to the subtasks A and B. Subtask A has two labels named as HOF and NOT (binary classification) [13] and subtask B has four labels named as HATE, OFFN, PRFN and NONE [21].

Number of comments in Training and Testing sets depicted in the table 2 and 3. It can be seen that majority of comments are not hate and offensive in the subtask B. It should be noted that this class imbalance problem should be handled while training a text classification model. Figure 2 shows the word cloud of top 100 most frequent words from the dataset.

Table 2
Distribution of Twitter comments in the training & testing dataset according to labels of subtask A

Labels	No of comments		Distribution of comments according to labels of subtask A
	Training set	Testing Set	
HOF	1856	423	50.1%
NOT	1852	391	49.9%
Total	3708	814	

Table 3
Distribution of Twitter comments in the training & testing dataset according to labels of subtask B

Labels	No of comments		Distribution of comments according to labels of subtask B
	Training set	Testing Set	
NONE	1852	414	50 %
PRFN	1377	293	36.5%
OFFN	321	82	9%
HATE	158	25	4.5%
Total	3708	814	

Figure 2: Word cloud of top 100 most frequent words the dataset

4. Proposed Approach

This section presents the approaches and their strategies for our submissions to the sentiment classification and the Hate Speech and Offensive Content Identification tasks.

4.1. Text pre-processing

Since the social media opinions are text comments from users in natural language, they tend to have lot of linguistic and grammatical errors. Pre-processing of text is considered an essential step when handling large scale social media text data [22]. One of the main objectives of pre-processing is to eliminate errors and to reduce the dimension of corpus or dictionary.

- **Removal of punctuations and special characters:** Punctuations and special characters such as {@ , ! , $, % , ^ , ' , " ,} were removed from the comments.
- **Tokenization:** Words in the comments were tokenized based on space character.
- **Stop word removal:** We eliminated very few of English stop-words such as and, this, above, he, she and so on. However Dravidian language specific stops words are not considered in this case.
- **Stemming:** Snow-Ball Stemmer technique was used to stem the words in the comments. We have used this stemming method only for HASOC dataset.
- **Spelling error correction:** In Dravidian-Codemix dataset, words with more than 3 consecutive redundant characters were cleaned by removing repetitive characters for e.g. **"**Thalivaaaaa**…"** word is converted into **"**Thaliva**".**
- **Others:** Usually, Twitter comments having hash tags(#), usernames (starts with @ and end with colon ' : '), URL's (starts with "https://") and 'RT' were eliminated from the comments [23].

4.2. Feature Representation

After preprocessing we ended up with 27986 and 8974 unique words in the Dravidian-Codemix and HASOC datasets correspondingly. Due to lack of standard NLP lexicon for the given Dravidian languages, we have directly processed the preprocessed text for feature extraction without performing high-level operations such as stemming or lemmatization. Feature representation is the process of converting textual data into meaningful numerical features to give as input into a learning algorithm. Here, two main approaches used for the feature representation are TF-IDF (for Machine Learning models) and Word Embedding (for Deep Learning models). Both of these representations are considered generic in nature where no language specific representation was utilized for this step.

4.2.1. TF-IDF

TF-IDF representation is generally used for training machine learning models. TF-IDF scheme is used to represent statistical frequency of the words in a sentence based on a corpus. If $\{\mathbf{w_1}, \mathbf{w_2}, \mathbf{w_3}, \ldots \mathbf{w_n}\}$ is represents the unique set of *n* tokenized words in the entire dataset, vector $\{\mathbf{v_{i1}}, \mathbf{v_{i2}}, \mathbf{v_{i3}}, ..\}$ represents the TF-IDF vector corresponding comment i, where $\mathbf{v_{ij}}$ is the statistical weight of word $\mathbf{w_j}$ for comment 'i' .

$$v_{ij} = Tf_{ij} * Idf_j$$

where Tf_{ij} = Term frequency of the j[th] term in comment 'i' and Idf_j = Inverse document frequency of the j[th] word. TF-IDF representation balances between both frequency of a word/term in a comment as well as importance or rarity of the terms in the entire dataset. To keep the feature representation in manageable high dimension we have used only unigram text representation. Since vocabulary of size (27986 words for Dravidian-Codemix and 8974 words for HASOC datasets correspondingly) is considered high-dimensional, we constructed TF-IDF matrix with top important 3000 and 2000 words from the Dravidian-Codemix and HASOC corpuses correspondingly.

4.2.2. Word Embedding

Word Embedding is yet another powerful text presentation strategy widely used in the recent times after the success of Deep Learning approaches. In word embedding, high-dimensional tokenized words from the corpus or dictionary are mapped into low-dimensional vectors of real numbers [24].

Dravidian-Codemix: Using our custom word embedding (supported by Keras python package) of 256 dimensions, which is learned using neural networks, every word in the corpus is represented using a fixed sized vector (256-dim).

HASOC: We are using the Glove Twitter Word Embedding [25] for the word representation. Glove is one of the pre-trained word-embedding for the twitter comments developed by Stanford NLP group in Stanford University. Glove model generally comes with dimensions such as 25, 50, 100 and 200. We used embedding model of 200 dimensions, which is learned using neural networks, every word in the corpus is represented using a fixed sized vector (200-dim). The 200-dimensional embedding model contains 1193515 word vectors. These word-level embeddings are further used in the Recurrent Neural Networks as input token while training them.

4.3. Text Classifiers

Following are the machine learning and deep leaning models used for text classification. Since both the datasets were imbalanced in terms of distribution of class labels, it was resolved by using class weightage when training a model. This is achieved by applying the weights of each class (i.e. distribution of each class in datasets) during training. For e.g. Positive class was assigned a weight as 68% or 0.68. For Dravidian-Codemix dataset, the final class weights were set as {0.11, 0.13, 0.68, 0.03, 0.05} for the classes Mixed feelings, Negative, Positive, Not-Tamil, Unknown-state correspondingly. In HASOC dataset, final class weights for subtask B were set as {0.50, 0.365, 0.09 and 0.045} for the classes NONE, PRFN, OFFN and HATE correspondingly

Decision Trees

Decision tree classifier is a rule-based learning model in which leaf nodes represent labels and other intermediate nodes represents a binary rule involving TF-IDF features. We have used entropy as a criterion for splitting nodes. The model was trained using without any regularization or pruning parameters. The model was trained using without any regularization or pruning parameters. For Dravidian-Codemix dataset, we ended up with a decision tree with depth of 851 and 6298 number of leave nodes. Decision tree trained for subtask A in HASOC dataset ended up with depth of 131 and 374 number of leaf nodes, respectively. Similarly, depth and number of leaf nodes obtained for the task B are 161 and 638 accordingly.

Random Forest

Random forest is one of the top performing models in Machine Learning and is a classic example of ensemble learning. Collection of randomly generated decision trees create a Random Forest where the final decision or labelling is based on voting of individual decision trees. Each tree utilizes the TF-IDF features for learning the rules. Here random forests with 10 (for Dravidian-Codemix) and 100 (for HASOC) decision trees were trained with Gini as a criterion for splitting nodes.

AdaBoost

AdaBoost is an acronym of Adaptive Boosting which is also one of the ensemble classifiers. It is mainly used for boosting the performance of a weak decision tree classifier. For Dravidian-Codemix, random forest with 100 decision trees were trained based on adaptive boosting with entropy as a criterion for splitting nodes. Learning rate of the model was kept as 1.0. values of minimum samples per leaf and minimum samples per split were set as 1 and 2, respectively. For all decision tree based models such as Decision trees, Random Forest and AdaBoost, minimum samples per leaf and minimum samples per split are set as default values of 1 and 2 correspondingly. Also, the parameters for maximum values of depth, features, samples, and leaf nodes in the trees were not restricted during the training of all these models. That is these models are indirectly encouraged to fit the training data well without any regularization.

Naïve Bayes

Naïve Bayes classifier is a probabilistic model which works based on the naïve assumptions i.e. strong independent assumptions between the features of sentences and sentiments. In general, Naïve

Bayes works with discrete features such as word counts of a dataset, however in practical it can also be used with TF-IDF features. This model gives the probability of sentiment classes based on the TF-IDF vectors of the dataset. We have used Multinomial Naïve Bayes and Bernoulli Naïve Bayes models for Dravidian-Codemix and HASOC datasets correspondingly.

Long Short Term Memory

Long Short Term Memory (LSTM) [26] is a type of deep learning model which is also popular variant of Recurrent Neural Network (RNN) modeling. LSTM has feed-back connections as well as feed-forward connections within. RNNs are mainly used for processing the sequential data such as speech, video, text etc. Unlike other RNN models, LSTM cells have their own memory to carry information even for lengthy sequence of tokens. Here for text classification, input to the model is sequence of embedding vector either from our custom word embedding (for Dravidian-Codemix) or from 200 dimensional Glove word embedding (for HASOC). Further we used one layer of Bi-directional LSTM with 128 nodes (64 nodes for HASOC) connected to final dense layers with 5 nodes (2 nodes for HASOC) having SoftMax optimization. We have compiled the model (Embedding layer + Bidirectional LSTM layer + Dense output layer) with cross entropy loss function and trained network with RMSProp optimizer (Adam optimizer for HASOC) for 50 epochs.

Gated Recurrent Unit

Gated Recurrent Unit (GRU) is a type of deep learning model which is another popular variant of Recurrent Neural Network (RNN) modeling which has a gating mechanism in RNN. GRU is also having feed-back connections as well as feed-forward connections like LSTM along with a forget gate. GRUs are mainly used for polyphonic music modeling, modeling of speech signal, etc. GRU gives better performance on tiny and less common data [28]. Here for hate speech identification, we input the model with sequence of embedding vectors from our Glove word embedding. Further we used Bi-directional GRU layer with 128 nodes connected with further two dense layers with 128 and 4 nodes having Relu and SoftMax activations accordingly. We have compiled the model with cross entropy loss function and trained network with Adam optimizer for 50 epochs.

Dravidian-Codemix: Aforementioned models have achieved accuracy more than 97% in the training set. In addition, we have also trained SVM and Bernoulli Naïve Bayes models which were not able to achieve good fitting to the training set.

HASOC: The machine learning models have achieved training set accuracies more than 90% and 80% for subtasks A and B respectively. In addition, we have also trained AdaBoost, simple RNN models which were not able to achieve good fitting for the two tasks.

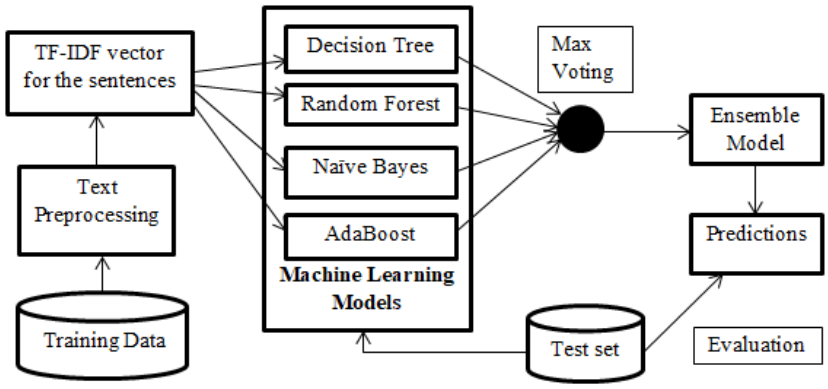

Figure 3: Architecture of our ensemble approach

Ensemble model

Finally, we have constructed an ensemble model by the maximum voting technique of previously generated four machine learning classifiers. Models such as Decision tree, Random Forest, Multinomial Naïve Bayes and AdaBoost were used in the Dravidian-Codemix task. Sample architecture of our ensemble approach used in Dravidian-CodeMix was shown in Figure 3. We can assume that SVM model is used instead of Adaboost in the figure 3.

5. Experimental Results and Discussion

Testing set submissions of Dravidian Code-Mix were evaluated using weighted average precision, recall and F1-score. HASOC's testing set submissions were evaluated using macro average F1-score. For implementations, we have utilized Python language and its packages such as Sklearn for pre-processing and training machine learning models and Keras for training deep learning models. The implementations of our submissions are available at https://github.com/Harikrishnancse/Dravidian-CodeMix-FIRE-2020 and https://github.com/Harikrishnancse/HASOC-2020 for Dravidian-Codemix and HASOC respectively.

5.1. Results of Dravidian-Codemix

Performance of our developed classifiers on testing dataset is depicted in table 4.

Table 4
Experiments results of proposed classifiers on the test data

Measures / Classifier	Weighted Average Precision (%)	Weighted Average Recall (%)	Weighted Average F1-Score (%)	Accuracy (%)
Decision Tree	55%	58%	56%	58%
Random Forest	57%	63%	59%	63%
Naïve Bayes	**59%**	**65%**	**61%**	65%
AdaBoost	54%	62%	57%	62%
LSTM	61%	68%	**61%**	**68%**
Ensemble	59%	66%	**60%**	**66%**

Our submission Multinomial Naïve Bayes classifier was ranked 5th at the leaderboard of Dravidian-CodeMix-FIRE2020 challenge. The model was selected based on weighted average F1 score, precision, and recall, where the model gives 61% F1-score on test data. We can see that our simple bidirectional LSTM model top precision and recall. Ensemble model also gives good results compared to other machine learning classifiers i.e. 60% F1-score and achieving same precision score as Naïve Bayes. On the other hand, LSTM model achieves overall best accuracy of 68% compared to other machine learning and ensemble models. Figure 4 depicts the confusion matrix of top performing models i.e. Naïve Bayes, Ensemble and LSTM for the testing set. Overall, we can see that the class "mixed feeling" is often confused with classes "positive" and "negative" by almost all models. This happens since the mixed feeling sentiment is closely similar to both positive and negative opinion. We can also see that less frequent classes "Non-Tamil" and "unknow-state" are confused with highly frequent class "positive". This is evidently showing the possible existence of class imbalance problem despite the measures taken for tacking them.

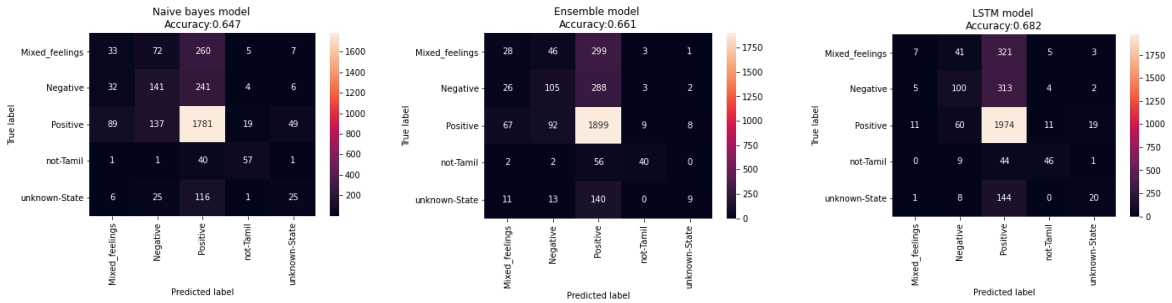

Figure 4: Confusion matrix of top performing models i.e. Naïve Bayes, Ensemble and LSTM.

5.2. Results of HASOC

Performance of our developed classifiers on testing dataset is depicted in table 5.

Table 5
Experiments results of proposed classifiers on the public test data of subtasks A and B

	Subtask A				Subtask B			
Measures / Classifier	Macro Average Precision	Macro Average Recall	Macro Average F1-Score	Accuracy	Macro Average Precision	Macro Average Recall	Macro Average F1-Score	Accuracy
Naïve Bayes	0.82	0.82	0.82	0.82	0.58	0.50	0.51	0.76
SVM	0.85	0.85	0.84	0.84	0.59	0.50	0.51	0.79
Decision Tree	0.83	0.83	0.82	0.82	0.46	0.45	0.46	0.72
Random Forest	0.85	0.84	0.83	0.84	0.60	0.46	0.45	0.80
Ensemble	0.86	0.86	0.86	0.86	**0.71**	**0.50**	**0.51**	0.81
LSTM	**0.88**	**0.88**	**0.88**	**0.88**	-	-	-	-
Bi-GRU	-	-	-	-	0.60	0.50	0.49	**0.82**

We can see that our simple LSTM model has achieved top precision and recall for subtask A i.e. 88% F1-score. Also, Ensemble model gives top score, precision and recall compared to other machine learning and deep learning classifiers i.e. 51% F1-score for subtask B. Next, Random Forest and SVM models give better score of 84% and 79% for both subtasks. Despite Bi-directional GRU gives high accuracy, it gives low F1-score and precision.

Table 6
Leaderboard results of best classifiers on the private test data of subtasks A and B

	Classifier	Macro F1-score	Place/Rank
Subtask A	LSTM	0.5002	14
Subtask B	Ensemble Model	0.2426	10

Table 6 shows the leaderboard results of our models tested on private test data. Our submission LSTM classifier achieved 14th rank and Ensemble classifier achieved 10th rank at the leaderboard for subtasks A and B correspondingly. These models are selected based on macro average F1 score, precision, and recall, where the models give 0.5002 macro average F1-score for subtask A and 0.2426 macro average F1-score for subtask B on private test data.

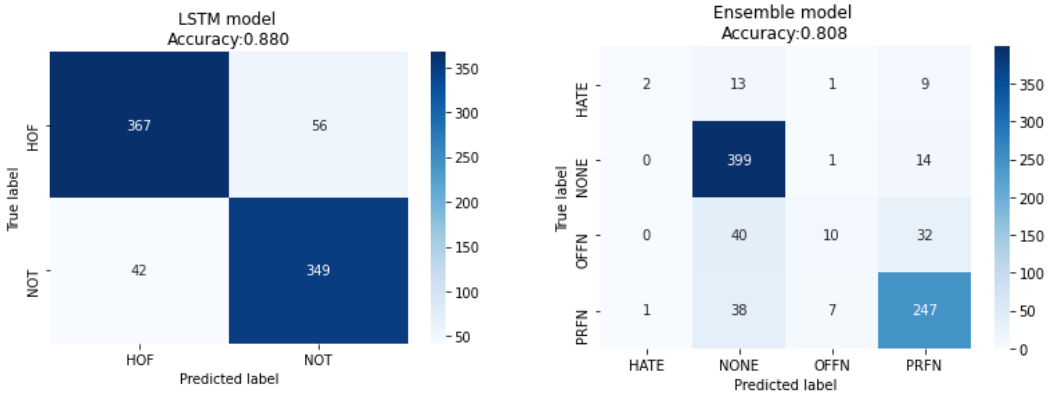

Figure 5: Confusion matrix of top performing models i.e. LSTM and Ensemble.

Figure 5 depicts the confusion matrix of top performing models i.e. Ensemble and LSTM for the public testing set.

6. Conclusion and Future Work

We presented our proposed text classification approaches based on Machine Learning and Deep Learning for the sentiment analysis of Tamil-English codemix and Hate Speech and Offensive Content Identification of English language. We processed and represented the raw comments using traditional language independent text preprocessing and document representation approaches. Based on the text representation we further exploited state-of-the-art machine learning algorithms such as Naïve Bayes, Decision tree, Random forest, AdaBoost, and deep Learning approaches such as bidirectional LSTM and bidirectional GRU. We have also made an ensemble model i.e. max voting of Machine learning classifiers. For Dravidian-CodeMix, we have achieved better results for Naïve Bayes, LSTM and Ensemble approach where these results are showing the difficulty of the sentiment analysis for Tamil-English codemix language. For HASOC, we have achieved better results for LSTM and Ensemble approach.

In future, we aim to develop language specific preprocessing techniques to eliminate the ambiguity and redundancy of words in the Dravidian Language codemix. We will also try to develop a custom multilingual pretrained Tamil-English Word Embeddings which uses external data to acquire better results. For hate speech detection, we aim to develop domain specific preprocessing techniques to eliminate the more ambiguity and redundancy in the English Language. Also, we will try to get the features from the hash tags and username annotations to develop the multi-level classification for identifying the victim. We will also try to use other popular Word Embedding such as Word2Vec and Sentence Embedding such as BERT, Doc2Vec to acquire better results.

References

[1] L. Yue, W. Chen, X. Li, W. Zuo, M. Yin, A survey of sentiment analysis in social media. Knowledge and Information Systems, 1-47 (2019).

[2] I. Chaturvedi, E. Cambria, R. E. Welsch, F. Herrera, Distinguishing between facts and opinions for sentiment analysis: Survey and challenges. Information Fusion, 44 (2018) 65-77.

[3] S. Phani, S. Lahiri, A. Biswas, Sentiment analysis of tweets in three Indian languages. In Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016), 2016, pp. 93-102.

[4] S. Seshadri, A.K. Madasamy, S.K. Padannayil, M.A. Kumar, Analyzing sentiment in Indian languages micro text using recurrent neural network. IIOAB J, 7, 2016, pp.313-318.

[5] B.R. Chakravarthi, R. Priyadharshini, V. Muralidaran, S. Suryawanshi, N. Jose, E. Sherly, J. P. McCrae. "Overview of the track on Sentiment Analysis for Davidian Languages in Code-Mixed Text" In Proceedings of the 12th Forum for Information Retrieval Evaluation, FIRE'20, 2020.

[6] B.R. Chakravarthi, R. Priyadharshini, V. Muralidaran, S. Suryawanshi, N. Jose, E. Sherly, J. P. McCrae. "Overview of the track on Sentiment Analysis for Davidian 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.

[7] B. R. Chakravarthi, V. Muralidaran, R. Priyadharshini, J. P. McCrae, Corpus creation for sentiment analysis in code-mixed Tamil-English text, In 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL), 2020, pp.202-210.

[8] B.R. Chakravarthi, N. Jose, S. Suryawanshi, E. Sherly, J. P. McCrae, A sentiment analysis dataset for code-mixed Malayalam-English, In 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL), 2020, pp.177-184.

[9] V. Uma, N. Kausikaa, Sentiment analysis of English and Tamil tweets using path length similarity based word sense disambiguation. IOSR J.(IOSR J. Comput. Eng.), 1, 2016, pp.82-89.

[10] N. Ravishankar, R. Shriram, Grammar rule-based sentiment categorization model for classification of Tamil tweets. International Journal of Intelligent Systems Technologies and Applications, 17(1-2) (2018) 89-97.

[11] Z. Waseem, D. Hovy, Hateful symbols or hateful people? predictive features for hate speech detection on twitter, In Proceedings of NAACL student research workshop, 2016, pp. 88-93.

[12] T. Davidson, D. Warmsley, M. Macy, I. Weber, Automated hate speech detection and the problem of offensive language, arXiv preprint arXiv:1703.04009, (2017).

[13] T. Mandl, S. Modha, G.K. Shahi, A.K. Jaiswal, D. Nandini, D. Patel, P. Majumder, J. Schäfer,(2020, December).Overview of the HASOC track at FIRE 2020: Hate Speech and Offensive Content Identification in Indo-European Languages. Working Notes of FIRE 2020 - Forum for Information Retrieval Evaluation.

[14] 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.

[15] N. Ousidhoum, Z. Lin, H. Zhang, Y. Song, D.Y. Yeung, Multilingual and multi-aspect hate speech analysis, arXiv preprint, arXiv:1908.11049, (2019).

[16] M.A. Bashar, R. Nayak, R, QutNocturnalHASOC'19: CNN for Hate Speech and Offensive Content Identification in Hindi Language, In Proceedings of the 11th annual meeting of the Forum for Information Retrieval Evaluation, 2019.

[17] V. Basile, C. Bosco, E. Fersini, N. Debora, V. Patti, F.M. Pardo, P. Rosso, M. Sanguinetti, Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter, In 13th International Workshop on Semantic Evaluation 2019, pp. 54-63.

[18] Schmidt, M. Wiegand, A survey on hate speech detection using natural language processing, In Proceedings of the Fifth International workshop on natural language processing for social media, 2017, pp. 1-10.

[19] Veena PV, Kumar MA, Soman KP. An effective way of word-level language identification for code-mixed facebook comments using word-embedding via character-embedding. In2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) 2017 Sep 13 (pp. 1552-1556). IEEE.

[20] Singh R, Choudhary N, Shrivastava M. Automatic normalization of word variations in code-mixed social media text. arXiv preprint arXiv:1804.00804. 2018 Apr 3.

[21] A.H. Razavi, D. Inkpen, S. Uritsky, S. Matwin, Offensive language detection using multi-level classification, In Canadian Conference on Artificial Intelligence, 2010, pp. 16-27.

[22] M. Anandarajan, C. Hill, T. Nolan, Text Preprocessing. In Practical Text Analytics, 2019,pp. 45-59.

[23] P.L. Teh, C.B. Cheng, W.M. Chee, Identifying and categorising profane words in hate speech, In Proceedings of the 2nd International Conference on Compute and Data Analysis, 2018, pp. 65-69.

[24] B.R. Chakravarthi, Leveraging orthographic information to improve machine translation of under-resourced languages (Doctoral dissertation, NUI Galway), 2020.

[25] 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.

[26] S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural computation, 9 (8) (1997) 1735-1780.

[27] 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.

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