Detecting Hate Speech and Offensive Content in English: A Model Using DistilBERT and A Model Based on FastText Strategies

Kongqiang Wang, Xiaobing Zhou*

School of Information Science and Engineering, Yunnan University, Kunming 650500, Yunnan, China.

Abstract

We use two advanced methods in Task HASOC[1] (2024) Task-1 Binary Classification in English. The first method is based on DistilBERT, which is the result of knowledge distillation based on BERT and is a lightweight and efficient pre-training model. The second one is data enhancement, carried out through synonym substitution, random insertion, and deletion to expand the training data set. The model mainly uses the GloVe word feature vector for the CLR (cycling learning rate) method and the sparse multi-head attention mechanism for supervised training. Two methods have achieved good results in this task, ranked 3rd in Task-1 HASOC 2024.

Keywords

DistilBERT, Data enhancement, CLR, Binary class Classification

1. Introduction

Social media today is a hotbed of curbing hate speech[2] and has emerged as a critical challenge for governments globally. People spend considerable time on social media like Facebook, Twitter, and Instagram. Studies suggest that most of the online content generated on these platforms contains abusive language. There is a need to develop adequate response mechanisms in order to find a balance between freedom of expression on one side, the ability to live without oppressive remarks on the other and a requirement for a robust technology to identify problematic content automatically[3].

HASOC [4] provides a forum for developing and testing text classification systems for various languages. It organized a shared task for FIRE 2024. The task is aimed at identifying hateful and offensive language in social media posts. The task is organized for two languages, English and Bengali, but we only conduct our investigation on the English dataset.

The following is a description of our research based on our two advanced solutions to the specific problem.

Task-1 Binary Classification in English: A task focused on hate speech and offensive language identification is offered for Hinglish. It is a coarse-grained binary classification in which participants are required to classify tweets into two classes, namely: hate and offensive (HOF) and non-hate and offensive (NOT).

- (NOT) Non Hate-Offensive - This post does not contain any hate speech, profane, offensive content.
- (HOF) Hate and Offensive - This post contains hate, offensive, and profane content.

Various ML and DL models [5] have been surveyed to check which produces more accuracy in detecting the various hate and offensive comments. But they didn't work well, so a pre-trained DistilBERT model was created and used, and it also inspired us to propose our model with the help of DistilBERT and another method using data enhancement to expand the train data set. The model mainly uses the GloVe word feature vector for the CLR (cycling learning rate) method and a sparse multi-head attention mechanism for supervised training to classify the HOF comments.

Forum for Information Retrieval Evaluation, December 12-15, 2024, India

*Corresponding author.

wangkongqiang60@gmail.com (K. Wang); zhouxb@ynu.edu.cn (X. Zhou)
https://github.com/WangKongQiang (K. Wang)

2. Exploratory Data Analysis

Table 1
Details of Dataset Provided by Organizer.

Details	Posts in Train Data	Posts in Test Data
Task-1 Binary Classification in English	Totall=NA	
Hate and Offensive posts (HOF)	NA	888
Non Hate and Offensive posts (NOT)	NA	

For Task-1, participants are allowed to use any external resources and datasets. The test dataset will be released later for the evaluation of models.

3. Graphical Representation

The following images are the word-cloud for various types of comments given in the dataset. It is a graphical representation of word frequency of different words used in each category of the comments on external resources and datasets.

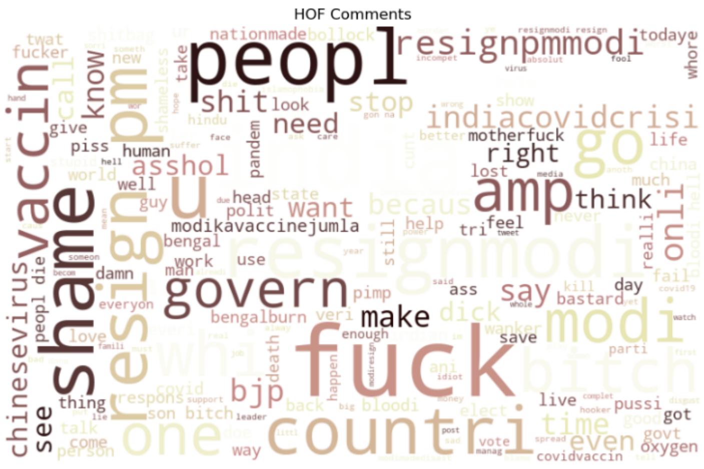


Figure 1: Word Cloud for HATE comments.

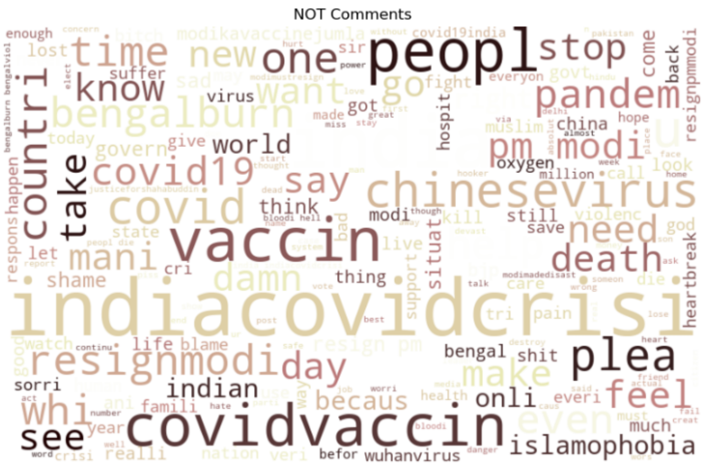


Figure 2: Word Cloud for NOT comments.

4. Methodology

4.1. Introduction

We give a general overview flow chart of the two models, in which the data preprocessing and data enhancement are similar, and the two methods are based on two different data sets. Both of which are related to this task. The two datasets and the model overview based on these two datasets will be introduced later.

4.2. Two Datasets

Table 2
Data set used in Method 1.

Details	Posts in Train Data	Posts in Valid Data
Task-1 Binary Classification in English	Totall=3843	
Hate and Offensive posts (HOF)	2501	577
Non Hate and Offensive posts (NOT)	1342	

Table 3
Data set used in Method 2.

Details	Posts in Train Data	Posts in Valid Data
Task-1 Binary Classification in English	Totall=3794	
Hate and Offensive posts (HOF)	1801	570
Non Hate and Offensive posts (NOT)	1993	

4.3. Pre-Processing

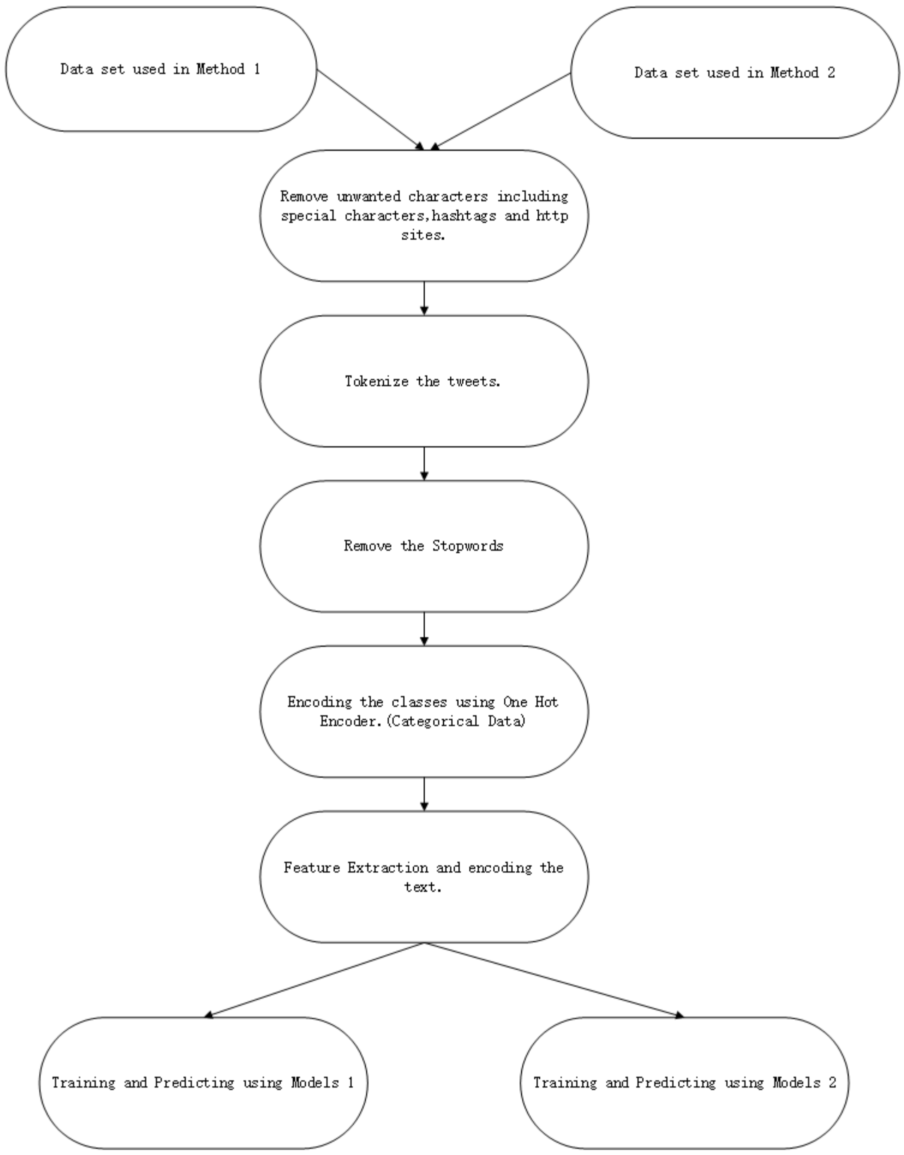
Natural language refers to a language that has formed and evolved naturally over a long period, such as Hindi and English and is commonly spoken and used. Natural language processing analyzes the meaning of natural language to enable computers to process the language. Natural language processing is applied in areas such as text classification, sentiment analysis, summarization, and text clustering. This processing includes three steps:

1. Text collection;
2. Text preprocessing;
3. Classification learning model.

In the first step (text collection), the texts to be processed are collected. The second step (text preprocessing) involves standardizing unstructured texts to increase the accuracy of natural language processing. The text collected contains many elements that are difficult to analyze, such as tags, references, and abbreviations. Most are expressed as if speaking in a carefree manner, in terms of the vocabulary or the structural order of the sentence. Therefore, after text processing, including changing the uppercase to lowercase, deleting special characters, tags, @'s, and removing stopwords, preprocessing is performed according to the requirement, and stemming is also done. Finally, a supervised learning model is established in the classification learning modeling stage, and training and prediction are performed using vectorized number-type data using TF-IDF. In this study, we use DistilBERT and another innovation model based on data enhancement for training and prediction and hate and offense detection.

4.4. Overall Flow

The diagram below depicts the flow of the experiment from pre-processing to classifying the text using state-of-the-art techniques.



Data set used in Method 1
Data set used in Method 2
Remove unwanted characters including special characters,hashtags and http sites.
Tokenize the tweets.
Remove the Stopwords
Encoding the classes using One Hot Encoder.(Categorical Data)
Feature Extraction and encoding the text.
Training and Predicting using Models 1
Training and Predicting using Models 2

Figure 3: Overall flow and methodology of the experiment.

4.5. Feature Extraction

4.5.1. TF-IDF

Initially, BoW was used for training. However, training the models was relatively inefficient owing to the fact that BoW doesn't consider term ordering and the rareness of the term. Hence, we use TF-IDF to overcome some of these drawbacks. The TF-IDF model contains information on the more important words and the less important ones, thus performing well on the ML models.

4.5.2. BERT Encoding

The ktrain2 library, which contains the Transformers API, is used to allow the use of any Hugging Face transformer model. And we use DistilBERT. The preprocess-train and preprocess test functions are used for the model name 'distilbert-base-uncased'. Word embeddings are generated from the transformer model, which is used for encoding. The ktrain is a lightweight wrapper for the deep learning library TensorFlow Keras to help build, train, and deploy neural networks and other machine learning models to make deep learning and AI more accessible and easier to apply.

4.5.3. GloVe Word Vector

GloVe (Global Vectors for Word Representation) is an unsupervised learning algorithm for obtaining vector representations of words. It is an extension of the Word2Vec model and helps to learn word vectors efficiently. The training is performed on aggregated global word-word co-occurrence statistics from the corpus, and the resulting representations showed interesting linear substructures of the word vector space.

Pennington et al. argue that the online scanning method used by the Word2vec algorithm is suboptimal because it does not take full advantage of statistics about word co-occurrence. Therefore, they developed a GloVe model combining the advantages of the Word2vec skip syntax model for word analogy tasks with a matrix decomposition method that can leverage global statistics. In classical vector space models, representations of words are developed by using matrix decomposition techniques such as Latent Semantic Analysis (LSA), which do an excellent job of using global text statistics but are not as good at capturing meaning and demonstrating it in tasks such as computational analogies as learning methods such as the Word2vec model.

For example, male/female relationships are automatically learned and represented by induced vectors, with "king - man + woman" resulting in vectors very close to "queen".

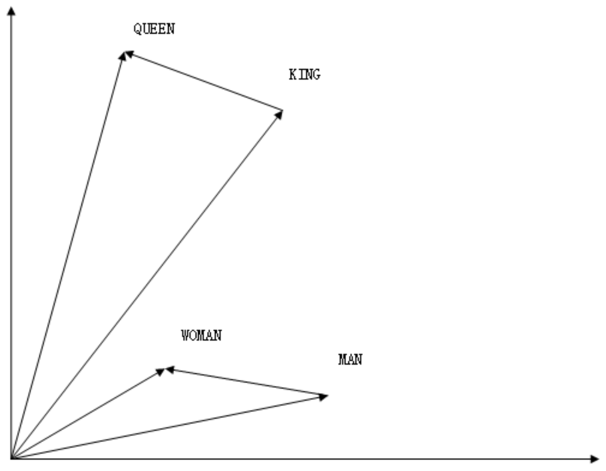


Figure 4: GloVe semantic analysis example.

The advantage of GloVe is that, unlike Word2vec, GloVe not only relies on local statistics (local context information for words), but combines global statistics (word co-occurrence) to obtain word vectors. GloVe can learn semantic relationships between derived words from the co-occurrence probabilities matrix.

5. Classifiers

5.1. Machine Learning Methods

For our experiment, we use many approaches[6]. First, a frequency dictionary is built to train the model and obtain the theta parameter for the sigmoid function that would better represent the data. The accuracy for this model came to 66%.

To improve the accuracy, we go ahead with TF-IDF Vectorizer, which is used to train a logistic regression model. The model is imported from the Scikit-learn Python package. This improved the accuracy of the model to about 80%.

To enable classification beyond the binary scope, we apply the K-Nearest Neighbor Classification technique. We use the Euclidean Distance as the criterion to determine the clusters that would be formed. This is used specifically for the task where there are two labels for classification, namely (HOF, NONE). The validation accuracy of this model came to about 75% for Task-1.

For the last model, we proceed with a Random Forest Classifier with an entropy criterion. Random Forest tends to behave better in the case of noisy data as well. This pushes the accuracy of our model to about 67% with Task-1.

Finally, we combine all models' predictions and take the majority of the prediction (mode) as the final say for Task-1.

Table 4
Validation accuracy results for ML Models Task-1 Binary Classification in English.

Model	Accuracy	Macro-F1	Macro-Precision	Macro-Recall
Logistic Regression	80%	74%	80%	73%
SVC	80%	77%	76%	77%
KNN	75%	69%	72%	68%
Random Forest	80%	77%	76%	77%

5.2. LSTM

Long Short-Term Memory (LSTM)[7] is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs. We use LSTM-based neural network classifiers for Task-1. We use word tokens, Embedding layer (256 dimensions), and input length (2500 words) as the inputs to the LSTM (64 units) layer with a dropout of 20% and recurrent dropout of 20%, softmax layer (for prediction) in the Keras toolkit. In this pipeline, we use binary cross entropy as a loss function, the Adam optimizer to optimize the parameters.

Table 5
Validation accuracy results for LSTM Model on Task-1 Binary Classification in English.

Model	Accuracy	Macro-F1	Macro-Precision	Macro-Recall
LSTM	78%	74%	76%	73%

5.3. DistilBERT

DistilBert[8] runs 60% is faster while preserving over 95% of BERT's performance. It also uses fewer parameters than BERT. It outperforms BERT and has now cemented itself as the model to beat for text classification and advanced NLP tasks. Implementing the DistilBERT pipeline is much easier than using any other pre-trained learning and thus helpful in performing transfer learning. Hence, we choose to use DistilBERT.

It contains 6 layers, 768 dimensions and 12 heads with 66 Million parameters and is pretrained on BookCorpus, a dataset consisting of 11,038 unpublished books and English Wikipedia. For the English dataset, we use a pre-trained 'distilbert-base-uncased' model for Task-1. The model is trained and validated with the validation dataset and finally tested on the test set given. For training, the max-len of the input sequence is chosen to be 500. If you set the max-length very high, you might face memory shortage problems during execution. The following results are inferred when testing the model on the validation dataset (15% split from the training dataset).

Table 6
Validation accuracy results for DistilBERT Model Task-1 Binary Classification in English.

Model	Accuracy	Macro-F1	Macro-Precision	Macro-Recall
DistilBERT	75%	69%	71%	68%

6. Two advanced methods

We use two advanced methods [2] in Task HASOC (2024) Task-1 Binary Classification in English. The first method is based on DistilBERT, which is the result of knowledge distillation based on BERT and is a lightweight and efficient pre-training model. The second one is data enhancement, which is carried out through synonym substitution, random insertion, random deletion, etc., to expand the train data set. The model mainly uses the GloVe word feature vector for the CLR (cycling learning rate) method and sparse multi-head attention mechanism for supervised training. Two methods have achieved good results in this task.

6.1. Proposed Model 1

This uses the DistilBERT model trained for Task-1. The motive behind using this model is its accuracy on the validation dataset and the model's reliability. The comments are to be classified as HOF or NOT comments. There are three steps combined to form the proposed model. The models are as follows:

1. A series of data set comments are preprocessed on the original data. For example, optionally shorten words to their stems, remove consecutive letters to the single letters at the end, remove stopwords, remove emojis, and Return a list of words.
2. Two Python libraries, 'nltk' and 'keras', are together used to transform the hate comments from the original data into HOF or NOT comments tokenizer.
3. Finally, the DistilBERT model, trained on the original data comments from the sample dataset, is used to classify the complete comments respectively.

The flow of the model is as follows: The test data is initially passed to the DistilBERT model. The DistilBERT model trained for Task-1 identifies if a comment is of hate or not. This is essentially a binary classification (HOF / NOT).

Above is a binary classification model that returns whether the given text is hate or not. The nonhate comments predicted by this model are labeled as NOT. And the remaining comments are to be classified as HOF (HATE and OFFN).

Now the hate comments in the test data as filtered by the DistilBERT model as a part of the proposed model alone are passed to two Python libraries, 'nltk' and 'keras', that transform the original data comments into Tokenizer, including HOF comments (HATE and OFFN) and NOT comments to be further classified. Lastly, the other original comments in the test dataset are passed to a DistilBERT model trained to classify a text as either HOF or NOT (trained using the training dataset provided).

6.2. Proposed Model 2

6.2.1. Data Enhancement

The following data enhancement methods are used:

1. Synonym replacement (Replace n words in the sentence with synonyms from wordnet);
2. Random deletion (Randomly delete words from the sentence with probability p);
3. Random swap (Randomly swap two words in the sentence n times);
4. Random insertion (Randomly insert n words into the sentence).

6.2.2. Cyclical Learning Rate

Learning rate (LR) is one of the most important hyperparameters in neural network training, and it is very important to train neural networks quickly and efficiently. In simple terms, LR determines how much our current weight parameters change in the direction of reducing losses. It seems simple enough. However, as many studies have shown, this step alone can have a profound impact on our training, and there is plenty of room for optimization. A technique called Periodic Learning Rate (CLR) is introduced here, which is a very new and simple idea for setting and controlling the size of LR during training.

This callback implements a cyclical learning rate policy (CLR). The method cycles the learning rate between two boundaries with some constant frequency. The amplitude of the cycle can be scaled on a per-iteration or per-cycle basis. This class has three built-in policies.

"triangular": A basic triangular cycle with amplitude scaling.

"triangular2": A basic triangular cycle that scales initial amplitude by half each cycle.

"exp_range": A cycle that scales initial amplitude by gamma(cycle iterations) at each cycle iteration.

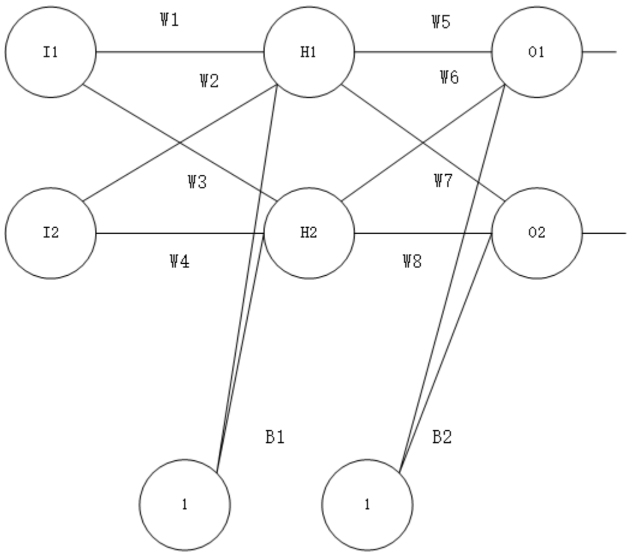


Figure 5: Technical weight calculation of periodic learning rate (CLR).

6.2.3. Sparse multi-head self-attention mechanism

Sparse attention is an optimized attention mechanism that can map a query vector and a set of key-value pairs to an output vector, but unlike single-headed and multi-headed attention, it does not calculate the similarity between the query vector and all key vectors, but only the similarity between the query vector and some key vectors. This reduces computation and memory consumption. The concept of sparse attention first appeared in 2018 [9], which proposed a long sequence generation model based on transformer. Sparse attention is used to process text sequences of more than 8,000 words.

Each element is associated only with elements whose relative distance is a multiple of rate, and with elements whose relative distance does not exceed that rate.

6.2.4. FastText

FastText is a fast text classification algorithm that has three major advantages over neural network-based classification algorithms:

1. FastText speeds up training and testing while maintaining high accuracy;
2. FastText does not need to pre-train the word vector, FastText will train the word vector itself;
3. FastText two important optimizations: Hierarchical Softmax, N-gram fastText model architecture.

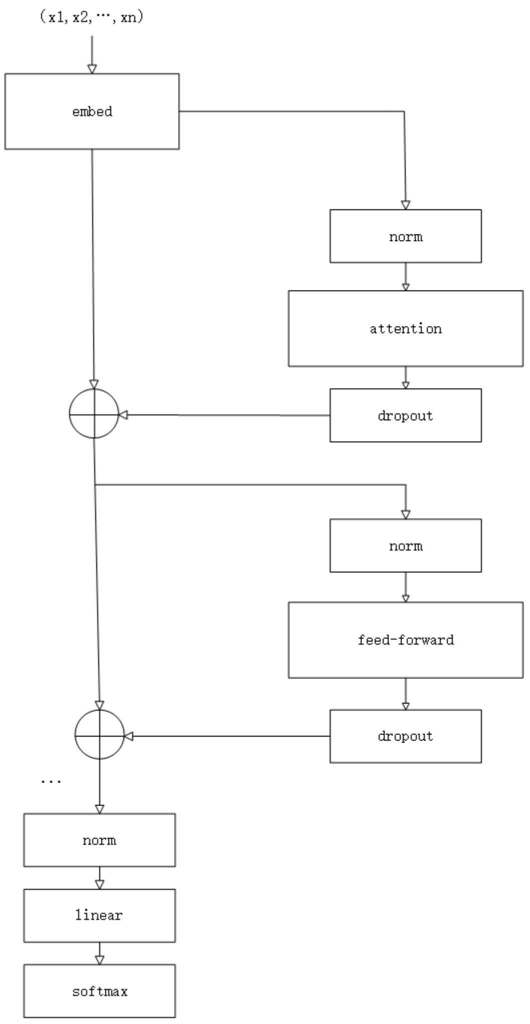


Figure 6: Sparse attention can map a query vector and a set of key-value pairs to an output vector.

The FastText model architecture is very similar to CBOW in word2vec. The difference is that FastText predicts labels while CBOW predicts intermediate words. That is, the model architecture is similar, but the model tasks are different. Let's take a look at the architecture of CBOW:

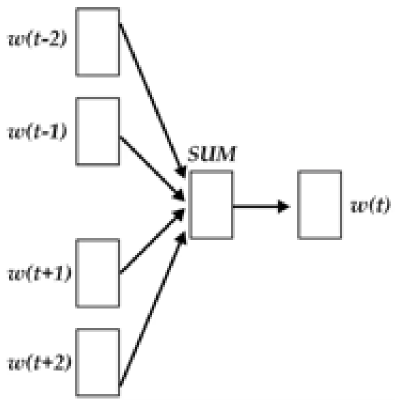


Figure 7: The architecture of CBOW content.

Word2vec trains a logistic regression model by turning the context into a multi-classification task, where the number of categories |V| is the overall size. In the usual text data, the thesaurus is as few as tens of thousands to as many as millions, so it is not practical to train the multi-classification logistic regression directly in training. Word2vec provides two optimization methods for large-scale multi-classification problems: negative sampling and hierarchical softmax. In optimization, negative sampling updates only a small number of negative classes, thus reducing the amount of computation. hierarchical softmax represents the dictionary as a prefix tree, and the path from root to leaf can be represented as a series of binary classifiers, reducing the complexity of computing multiple classes simultaneously from the height of |V| to the tree.

FastText model architecture: where x1,x2,... xN-1,xN represents an n-gram vector in a text, where each feature is the mean of the word vector. This is similar to CBOW, which was mentioned earlier, where CBOW uses context to predict the central word. Whereas here, all N-grams are used to predict the specified category.

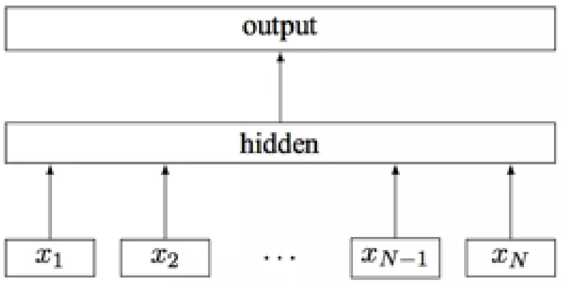


Figure 8: N-gram is used to predict the specified category.

7. Result

The ML model's accuracy is improved using K-Folding Technique with 10 splitting iterations in the cross-validator. Even though the validation accuracy of the ML models is decently high, the pre-trained DistilBERT model proves to be most accurate on the test data on submission with an excellent 77.67% accuracy score for Task-1. Our proposed other model gives the next best prediction for Task-1 with approximately 80% accuracy.

The ML model does not perform well for Task 1, a possible reason could be that the model could not differentiate between hateful and offensive posts properly. Data augmentation, exploring other pre-trained models such as XLNet, ERNIE, RoBERTa, and considering POS tags combined with n-grams to give an extra set of feature space could be the scope of improvement to solve this problem.

Table 7
Table : Overall validation accuracy for our proposed model 1.

Model	Accuracy	Macro-F1	Macro-Precision	Macro-Recall
Proposed Model 1	78%	73%	75%	72%

Table 8
Table : Overall validation accuracy for our proposed model 2.

Model	Accuracy	Macro-F1	Loss
Proposed Model 2	0.8172	0.8172	0.4779

8. Conclusion

In this paper, several machine learning and deep learning approaches have been used to detect hate speech and offensive language content, and the models have been compared. Several techniques have been employed to increase accuracy. Our proposed two-step model achieved good results compared to its simplicity, second only to the pre-trained DistilBERT model. We believe with proper feature extraction and data augmentation techniques, these will be able to improve our proposed model.

9. Acknowledgments

Thanks to the support of the official open source packages nltk and keras, as well as the release of pre-training distillation version Bert. We can achieve good results with simple fine-tuning after the model has been pre-trained.

10. Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] N. Raihan, K. Ghosh, S. Modha, S. Satapara, T. Gaur, Y. Dave, M. Zampieri, S. Jaki, T. Mandl, Overview of the HASOC Track at FIRE 2024: Hate-Speech Identification in English and Bengali, in: K. Ghosh, T. Mandl, P. Majumder, D. Ganguly (Eds.), Forum for Information Retrieval Evaluation (Working Notes) (FIRE 2024) December 9-13, Gandhinagar, India, CEUR-WS.org, 2024.

[2] S. Modha, T. Mandl, P. Majumder, D. Patel, Tracking hate in social media: Evaluation, challenges and approaches (2020). URL: https://link.springer.com/article/10.1007/s42979-020-0082-0.

[3] J.-C. Mensonides, P.-A. Jean, A. Tchechmedjiev, S. Harispe, Imt mines ales at hasoc 2019: Automatic hate speech detection (2019). URL: http://ceur-ws.org/Vol-2517/T3-13.pdf.

[4] K. Ghosh, N. Raihan, S. Modha, S. Satapara, T. Gaur, Y. Dave, M. Zampieri, S. Jaki, T. Mandl, Overview of the HASOC Track at FIRE 2024: Hate-Speech Identification in English and Bengali, in: FIRE '24: Proceedings of the 16th Annual Meeting of the Forum for Information Retrieval Evaluation. December 9-13, Gandhinagar, India, Association for Computing Machinery (ACM), New York, NY, USA, 2024.

[5] A. Parikh, H. Desai, A. S. Bisht, Da master at hasoc 2019: Identification of hate speech using machine learning and deep learning approaches for social media post (2019). URL: http://ceur-ws.org/Vol-2517/T3-18.pdf.

[6] H. A. Nayel, S. H. L., Deep at hasoc2019 : A machine learning framework for hate speech and offensive language detection (2019). URL: http://ceur-ws.org/Vol-2517/T3-21.pdf.

[7] M. Yi, Myung, J. Lim, H. Ko, J. Shin, Method of profanity detection using word embedding and lstm (2021). URL: https://doi.org/10.1155/2021/6654029.

[8] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter (2020). URL: https://arxiv.org/pdf/1910.01108.pdf.

[9] R. Child, S. Gray, A. Radford, I. Sutskever, Generating long sequences with sparse transformers, Machine Learning (2019). URL: https://arxiv.org/abs/1904.10509. doi:https://doi.org/10.48550/arXiv.1904.10509.