

# Classification of Hate Speech and Offensive Content using an approach based on DistilBERT

Swetha Saseendran, Sudharshan R., Sreedhar V. and Sharan Giri

*Sri Sivasubramaniya Nadar College of Engineering, Chennai, Tamil Nadu, India*

## Abstract

This paper describes the research that our team, 'Binary Beings', did on the shared task HASOC, conducted by FIRE-2021, which involves identification of hate and offensive language in various comments on social media. Our task is divided into two hierarchical sub-tasks on an English dataset. We employed and compared various Machine Learning, Deep Learning techniques and used pre-trained models to understand which model is most accurate in predicting the classes. Our best model [DistilBERT] obtained a Macro F1 score of 74.91% for SubTask-A and 57.65% for SubTask-B.

## Keywords

Machine Learning, LSTM, DistilBERT, Cross-Validation, TF-IDF, Multi-class Classification

## 1. Introduction

Social media today is a hotbed of curbing hate speech. This has emerged as a critical challenge for governments globally<sup>0</sup>. People are spending a considerable amount of time on social media like Facebook, Twitter, Instagram. Studies suggest that most of the online content generated on these platforms contain 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.

HASOC provides a forum for developing and testing text classification systems for various languages. It organized a shared task for FIRE 2021. The task is aimed at identifying hateful and offensive language in social media posts. The task was organized for two languages namely English and Hindi, but we conducted our investigation only on the English dataset. [8]

---


*Forum for Information Retrieval Evaluation, December 13-17, 2021, India*

✉ swetha18183@cse.ssn.edu.in (Swetha Saseendran); sudharshan18173@cse.ssn.edu.in (Sudharshan R.); sreedhar18161@cse.ssn.edu.in (Sreedhar V.); sharan18141@cse.ssn.edu.in (Sharan Giri)

🌐 <https://github.com/swetha4444> (Swetha Saseendran); <https://github.com/exploring-curiosity> (Sudharshan R.); <https://github.com/sreedhr92> (Sreedhar V.); <https://github.com/sharan0276> (Sharan Giri)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

---

There were two tasks given

**Task 1:** Identification and discrimination of hate, offensive and profane content from the post.

**Task 2:** Identification of conversational hate-speech in code-mixed languages.

Of these we chose to do Task 1. The problem statement we chose consists of hierarchical two-subtasks:

- **SUB-TASK A:** Identification of hate and offensive language, posts are classified as follows:
  - Hate and Offensive(HOF)
  - Not Hate and Offensive(NOT)
- **SUB-TASK B:** Categorization of Hate and Offensive, posts are further categorized as:
  - Hate Speech(HATE)
  - Offensive(OFFN)
  - Profane(PRFN)

Various ML and DL models have been surveyed to check which produces more accuracy in detecting the various Hate and Offensive comments. A pre-trained DistilBERT model is also used and it also inspired us to propose our own model with the help of DistilBERT and other Profanity Detection libraries (Profanity filter and better-profanity) for classifying the HOF comments detected from Sub-Task A, as HATE, OFFN or PRFN in Sub-Task B.

## 2. Exploratory Data Analysis

The datasets provided were annotated in a hierarchical fashion as shown below: Table 1: .Class Distribution of the data across sub-tasks

Table 1. Details of Dataset Provided

Details	Posts in Train Data	Posts in Test Data
<b>SUB-TASK A</b>	<i>Total = 3843</i>	
Hate and Offensive posts (HOF)	2501	1281
Non Hate and Offensive posts (NOT)	1342	
<b>SUB-TASK B</b>	<i>Total = 3843</i>	
Hate (HATE)	683	
Offensive (OFFN)	622	1281
Profane (PRFN)	1196	
NONE	1342	

In the training data for Subtask-A, 65% of the data are hate comments (HOF) and about 35% are non-hate comments (NOT). For Subtask-B, the 65% hate comments were classified as HATE (18% of the HOF comments), OFFN (16% of the HOF comments) and PRFN (31% of the HOF comments).

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

<sup>0</sup> <https://www.hindustantimes.com/analysis/it-is-time-to-regulate-hate-speech-on-social-media/story-x2JfnAcZ4mh404CM2wQLpO.html>



Fig 1: Word Cloud for HATE comments

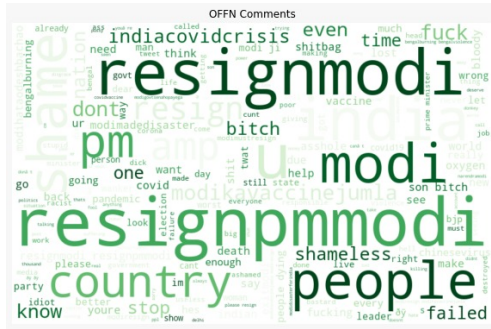


Fig 2: Word Cloud for OFFN comments



Fig 3: Word Cloud for PRFN comments

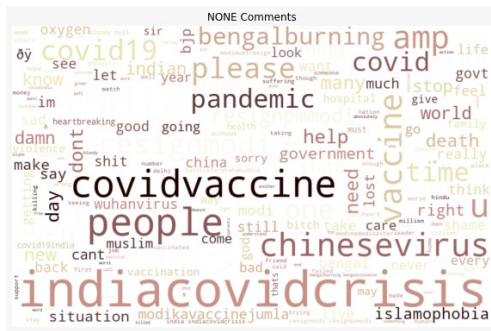


Fig 4: Word Cloud for NONE comments

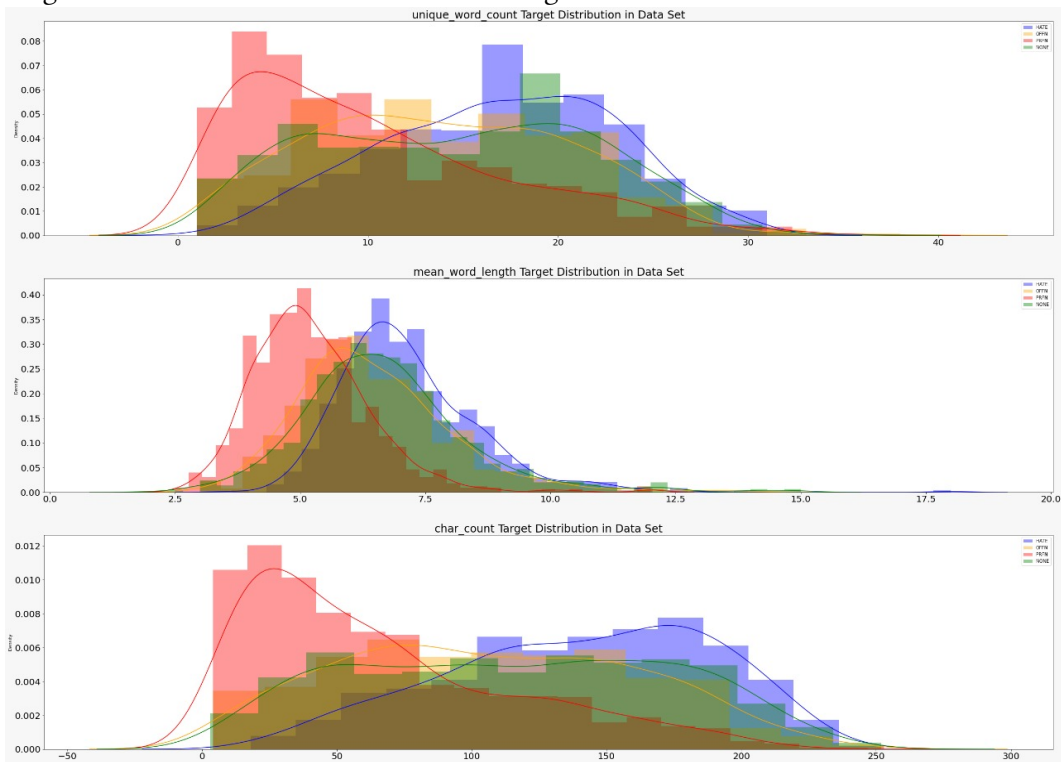


Fig 5: Meta Data Analysis of the training data

---

This explains the meta data analysis for training data. It summarizes basic information about data, making finding & working with particular instances of data easier. The diagram shows the distribution of unique words, the mean word length and character count of each type of comment over the training data set.

### 3. Methodology

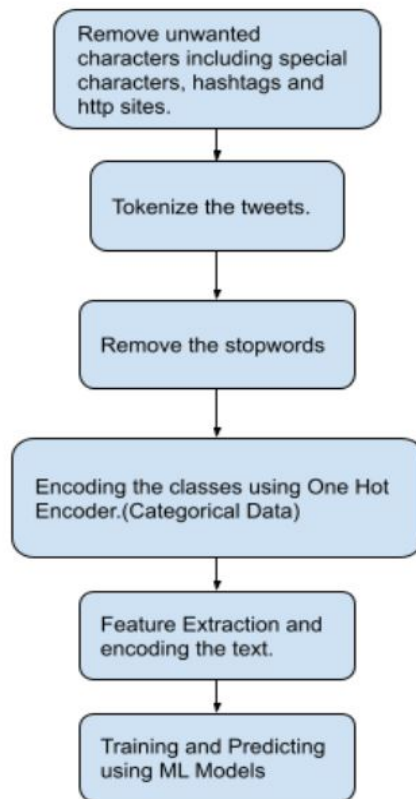


Fig 6: Overall flow and methodology of the experiment

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

#### 3.1. 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. Machine learning model

In the first step (text collection), the texts to be processed are collected. The second step (text preprocessing) involves the standardization of 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, abbreviations. Most are expressed as if speaking in a care-free 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, in the machine learning modelling stage, a supervised learning model is established, and training and prediction are performed using vectorized number-type data using TF-IDF. In this study, we use the LSTM, BERT and other basic ML models for training and prediction, for hate, offense and profanity detection.

## **3.2. Feature Extraction**

### **3.2.1. TF-IDF**

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

### **3.2.2. BERT Encoding**

The `ktrain`<sup>2</sup> library is used which contains the Transformers API, that allows the use of any Hugging Face transformers model. And we used DistilBERT. The `preprocess-train` and `preprocess-test` functions were used for the model name 'distilbert-base-uncased'. Word embeddings are generated from the transformer model which is used for encoding.

`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 and to make deep learning and AI more accessible and easier to apply.

## **3.3. Classifiers**

### **3.3.1. ML Methods**

For our experiment we used 3 approaches. First, a Frequency dictionary<sup>1</sup> was built, with which we 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 went ahead with TF-IDF Vectorizer, which was used to train a logistic regression model. The model was imported from the Scikit-learn python package. This improved the accuracy of the model to about 80%.

<sup>1</sup> <https://towardsdatascience.com/sentiment-analysis-using-logistic-regression-and-naive-bayes-16b806eb4c4b>

For the next two models, we used a cross-validation approach with ten folds to train different models and saved the model with the best accuracy score. This validation feature was imported from the scikit-learn model-selection package. For the second approach, we went ahead with an SVM model. Given the task was a classification, SVM seemed to be an easier and efficient way to classify the tweet sentiments. The best model that was saved had an accuracy of about 80%. [5]

To enable classification beyond binary scope, we went ahead to apply the K0.5 Nearest Neighbor Classification technique. We used the Euclidean Distance as the criterion to determine the clusters that would be formed. This was used specifically for Task B where there were four labels for classification namely (PRFN, OFFN, HATE, NONE). The validation accuracy of this model came to about 75% for Sub-Task A and 64% for Sub-Task B.

The last model, we went ahead with Random Forest Classifier with entropy criterion. Random Forest tends to behave better in the case of noisy data as well. This pushed the accuracy of our model to about 67% with SubTask A.

Finally, we combined the predictions of all models together and took the majority of the prediction (mode) as the final say for Sub-Task B.

Table 2: Validation accuracy results for ML Models Sub-Task A :

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%

Table 3: Validation accuracy results for ML Models Sub-Task B :

Model	Accuracy	Macro-F1	Macro-Precision	Macro-Recall
SVC	60%	52%	52%	53%
KNN	57%	43%	49%	46%
Random Forest	58%	49%	50%	50%

### 3.3.2. LSTM

Long Short-Term Memory (LSTM) 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 used LSTM based neural network classifiers for both Sub-Task A and Sub-Task B. We used word tokens, Embedding layer (256 dimensions), input length (2500 words) as the inputs to the LSTM (64 units) layer with dropout of 20% and recurrent dropout of 20%, softmax layer (for prediction) in the keras toolkit. In this pipeline, we used binary-crossentropy as loss function, the Adam optimizer to optimize the parameters.

<sup>2</sup> <https://pypi.org/project/ktrain/>

Fig 7 : LSTM model for Sub-Task A

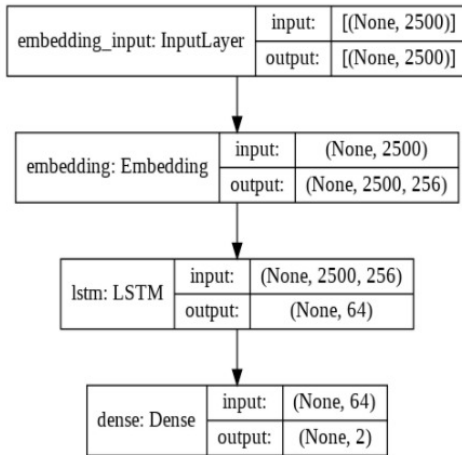


Fig 8: LSTM model for Sub-Task B

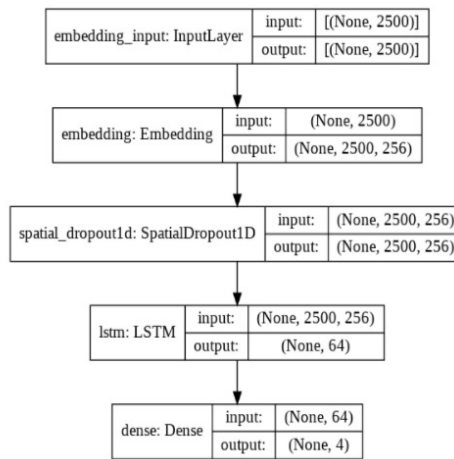


Table 4: Validation accuracy results for LSTM Model:

Task	Accuracy	Macro-F1	Macro-Precision	Maro-Recall
Subtask - A	78%	74%	76%	73%
Subtask - B	60%	17%	19%	21%

### 3.3.3. DistilBERT

Distilbert runs 60% faster while preserving over 95% of BERT's performance. It also uses less parameters than BERT. It outperformed BERT and has now cemented itself as the model to beat for not only text classification, but also advanced NLP tasks. Implementation of DistilBERT pipeline is much easier than using any other pre-trained learning and thus helpful in performing transfer learning. Hence, we chose to use DistilBERT.<sup>3</sup>

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<sup>3</sup>. For the English dataset, we used a pre-trained 'distilbert-base-uncased' model for both Sub-Task A and Sub-Task B. The model was trained and validated with the validation dataset and finally tested on the test set given. For training the max-len of the input sequence was chosen to be 500. If you set the max-length very high, you might face memory shortage problems during execution. On testing the model on validation dataset (15% split from training dataset) the following results were inferred:

Table 5: Validation accuracy results for DistilBERT Model:

Task	Accuracy	Macro-F1	Macro-Precision	Maro-Recall
Subtask - A	78%	73%	75%	72%
Subtask - B	61%	52%	55%	55%

<sup>3</sup> [https://huggingface.co/transformers/model\\_doc/distilbert.html](https://huggingface.co/transformers/model_doc/distilbert.html)

### 3.3.4. Proposed Model Task B

This uses the DistilBERT model trained for Subtask-A. The motive behind using this model is its accuracy on the validation dataset and the model's reliability. The HOF comments are to be further classified as PRFN, OFFN or HATE comments.

There are three separate models combined to form the proposed model. The models are as follows:

1. The DistilBERT model trained for Subtask-A identifies if a comment is of hate or not. This is essentially a binary classification (HOF/NOT).
2. Two python libraries 'profanityfilter'<sup>4</sup> and 'better-profanity'<sup>5</sup> are together used to classify the hate comments from the above model (1) into PRFN or other hate comments.
3. Finally, another DistilBERT model, trained on the OFFN and HATE comments from the sample dataset, is used to classify the remaining comments respectively.

The flow of the model is as follows: The test data is initially passed to DistilBERT model 1. above, which is a binary classification model that returns if the given text is hate or not. The non hate comments predicted by this model are labelled as NONE. And the hate comments are to be further classified as HATE, PRFN and OFFN.

Now the hate comments in the test data as filtered by the first model as a part of the proposed model alone is passed to two python libraries 'profanityfilter'<sup>4</sup> and 'better-profanity'<sup>5</sup> that classify the hate comments into PRFN hate comments and other hate comments (HATE and OFFN) to be further classified. Lastly, the other hate comments in the test dataset is passed to a DistilBERT model trained to classify a text as either HATE or OFFN (trained using the training dataset provided), which will further classify the text as OFFN and HATE and ultimately, all the text in the test dataset are now classified as HATE,OFFN,PRFN or NONE.

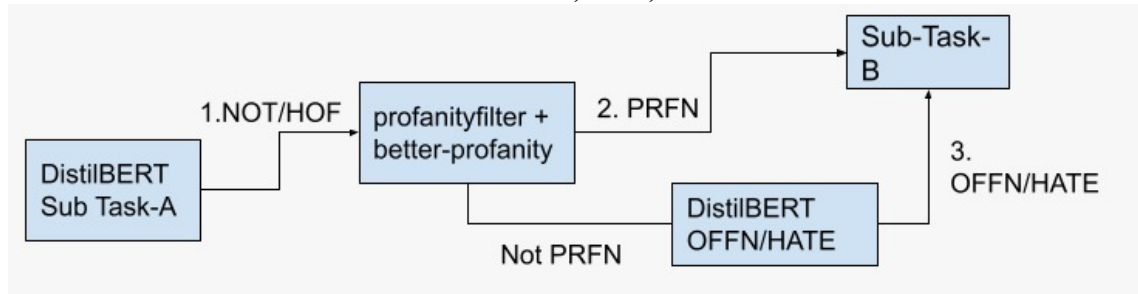


Fig 9: Proposed Model Block Diagram

Table 6: Validation accuracy results for each model used in our proposed model:

Model	Accuracy	Macro-F1	Macro-Precision	Macro-Recall
1. DistilBERT Subtask-A	78%	73%	75%	72%
2. Profanity filter + better-profanity	92%	48%	50%	46%
3. DistilBERT for OFFN/HATE classification	79%	79%	79%	79%

<sup>4</sup> <https://pypi.org/project/profanity-filter/>



Table 7: Overall validation accuracy for our proposed model:

Accuracy	Macro-F1	Macro-Precision	Macro-Recall
63%	59%	61%	58%

## 4. Result

The ML model’s accuracy improved using K-Folding Technique with 10 splitting iterations in the cross-validator. Even though the validation accuracy of the ML models were decently high, the pre-trained DistilBERT model proved to be most accurate on the test data on submission with a good 77.67% accuracy score for Sub-Task A and around 65% accuracy for Sub-Task B. Our proposed model gave the next best prediction for Sub-Task B with approximately 60% accuracy. The ML models did not perform well when for the Sub-Task B, a possible reason for it could be that the model could not differentiate between profane, 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.

Fig 10: Test Data Accuracies for Sub-Task A

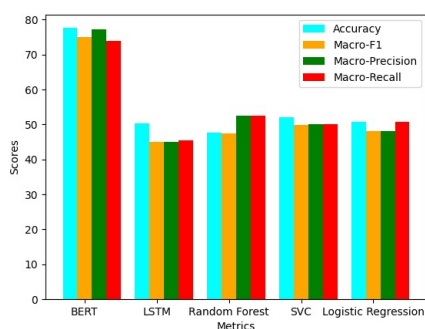


Fig 11: Test Data Accuracies for Sub-Task B

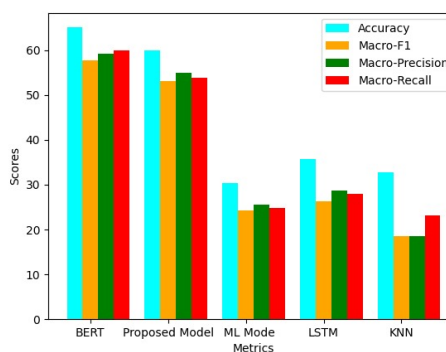


Table 8: Results obtained for Sub-Task A

Model	Accuracy	Macro-F1	Macro-Precision	Macro-Recall
DistilBERT	77.67%	74.91%	77.30%	74.91%
LSTM	50.35%	45.08%	45.06%	45.48%
Random Forest	47.77%	47.54%	52.86%	52.61%
SVM	52.06%	49.96%	50.04%	50.05%
Logistic Regression	50.66%	48.13%	48.18%	48.14%

<sup>5</sup> <https://pypi.org/project/better-profanity/>

Table 9: Results obtained for Sub-Task B

Model	Accuracy	Macro-F1	Macro-Precision	Macro-Recall
DistilBERT	65.1%	57.65%	59.90%	57.65%
Proposed Model	59.95%	53.04%	54.98%	53.91%
ML Overall Mode	30.44%	24.25%	25.57%	24.83%
LSTM	35.75%	26.34%	28.66%	27.98%
KNN	32.86%	18.58%	18.58%	23.19%

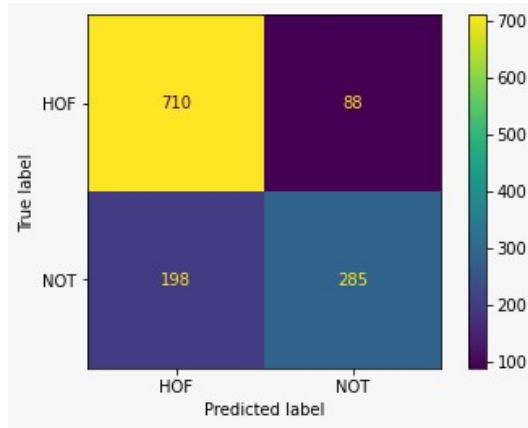


Fig 12: Sub-Task A: Confusion Matrix DistilBERT

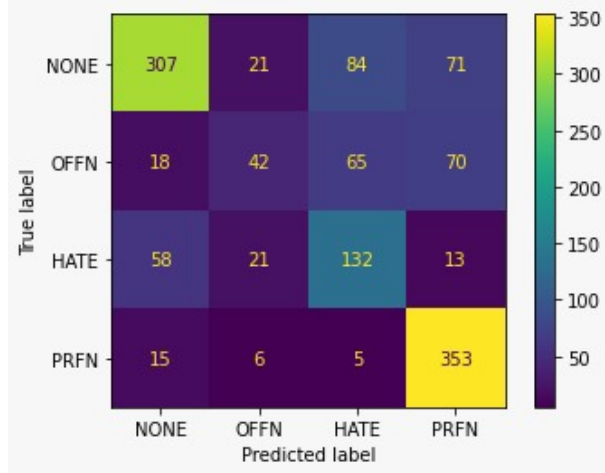


Fig 13: Sub-Task B: Confusion Matrix DistilBERT

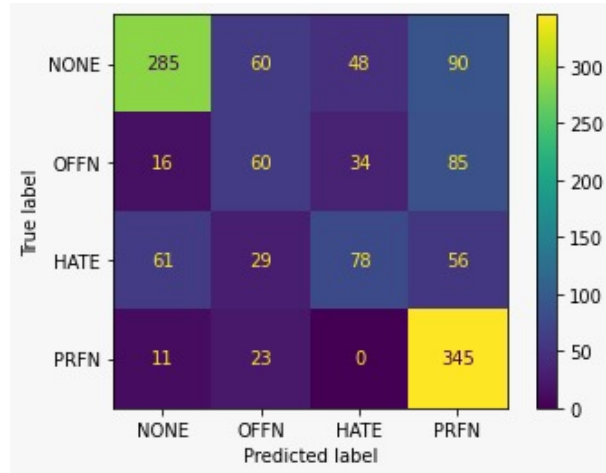


Fig 14: Sub-Task B: Confusion Matrix Proposed Model

## 5. Conclusion

In this paper, several machine learning and deep learning approaches have been used for detecting hate speech and offensive language content and the models have been compared. Several attempts on various techniques to increase the accuracy have been employed. Our proposed model achieved good results compared to its simplicity, second only to the pretrained DistilBERT model. We believe with proper feature extraction and data augmentation techniques, we will be able to improve our proposed model.

## References

- [1] Victor Sanh, Lysandre Debut, Julien Chaumond, Thomas Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL: <https://arxiv.org/pdf/1910.01108.pdf>.
- [2] Apurva Parikh , Harsh Desai , and Abhimanyu Singh 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>.
- [3] Vandan Mujadia, Pruthwik Mishra, Dipti Misra Sharma, IIIT-Hyderabad at HASOC 2019: Hate Speech Detection, 2019. URL: <http://ceur-ws.org/Vol-2517/T3-12.pdf>.
- [4] Moungho Yi, Myung, Jin Lim, Hoon Ko, and JuHyun Shin, Method of Profanity Detection Using Word Embedding and LSTM, 2021. URL: <https://doi.org/10.1155/2021/6654029>.
- [5] 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>.
- [6] Jean-Christophe Mensonides, Pierre-Antoine Jean, Andon Tchechmedjiev, and S ´ebastien Harispe, IMT Mines Ales at HASOC 2019: Automatic Hate Speech Detection, 2019. URL: <http://ceur-ws.org/Vol-2517/T3-13.pdf>.

- 
- [7] S. Modha, T. Mandl, G.K. Shahi, H. Madhu, S. Satapara, T. Ranasinghe, M. Zampieri, Overview of the HASOC Subtrack at FIRE 2021: Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages and Conversational Hate Speech, in: FIRE 2021: Forum for Information Retrieval Evaluation, Virtual Event, 13th-17th December 2021, ACM, 2021.
- [8] T. Mandl, S. Modha, G.K. Shahi, H. Madhu, S. Satapara, P. Majumder, J. Schäfer, T. Ranasinghe, M. Zampieri, D. Nandini, A. K. Jaiswal, Overview of the HASOC subtrack at FIRE 2021: Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages, in: Working Notes of FIRE 2021 - Forum for Information Retrieval Evaluation, CEUR, 2021. URL: <http://ceur-ws.org/>.
- [9] S. Jaki, T. De Smedt, M. Gwózdź, R. Panchal, A. Rossa, G. De Pauw, Online hatred of women in the Incels. me forum: Linguistic analysis and automatic detection. *Journal of Language Ag-gression and Conflict*, 7(2), 240-268., 2019. URL: <http://www.organisms.be/downloads/incels.pdf>.
- [10] W. Yin , A. Zubiaga, Towards generalisable hate speech detection: a review on obstacles and solutions, 2021. URL: <https://doi.org/10.7717/peerj-cs.598>.
- [11] 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>.