Hate Speech Detection

G Mukund, Hrishikesh Makwana, Somshuvra Basu

April 2023

1 Introduction

Hate speech detection is the process of identifying language that is used to express hatred, intolerance, or prejudice against a particular group or individual based on their race, ethnicity, religion, gender, sexual orientation, or other personal characteristics.

This problem of hate speech detection involves developing algorithms and models that can automatically detect hate speech in various forms of communication, such as social media posts, comments, and messages. This is a challenging task because hate speech can be expressed in many different ways and can be difficult to distinguish from other forms of speech.

To address this problem, various techniques such as natural language processing (NLP), machine learning, and deep learning can be used. These techniques involve training algorithms on large datasets of labeled hate speech and non-hate speech data, using features such as the use of specific words, phrases, and contextual information to identify hate speech, a few of which are demonstrated in this project.

Hate speech detection has become an important issue in recent years, as hate speech can have harmful effects on individuals and communities. By developing effective hate speech detection algorithms, we can better protect individuals from the negative impacts of hate speech and promote a more inclusive and respectful society.

This Project mainly deals with the Hate Speech Detection from Tweets.

2 Dataset Description

1. The Kaggle Twitter Sentiment Analysis dataset provided by Analytics Vidhya is a collection of tweets that have been labeled as either "hateful" or "non-hateful". The dataset contains over 31,000 tweets in total, with approximately 7.014% tweets labeled as hateful and the rest labeled as non-hateful.

2. The goal of this project is to develop a model that can accurately classify tweets as hateful or non-hateful based on the text content of the tweet.

3. The dataset has been preprocessed to remove URLs, user mentions, and other irrelevant information. It also includes a binary label indicating whether the tweet is hateful (1) or non-hateful (0).

3 Exploratory Data Analysis and Pre-Processing

1. We first check the distribution of the dataset, which appears to be highly imbalanced as non-hateful tweets are much higher in number than tweets labeled as hateful.

2. We understand how the dataset is built upon by analysing different visualisation techniques.

3. For pre-processing we begin by removing the unnecessary components such as symbols, tags, excape characters, emojis etc.

4. Next we remove the punctuation involved and convert the string in its lowercase form.

5. We then remove the stopwords present alongwith user.

6. Finally we lemmatize the the tweets.

4 RoBERTa Model

1. RoBERTa (Robustly Optimized BERT Approach) is a large-scale, pre-trained language model that was introduced in 2019 by Facebook AI Research. It is based on the same architecture as BERT (Bidirectional Encoder Representations from Transformers), but with several modifications and improvements.

2. RoBERTa is pre-trained on a large corpus of text using a masked language modeling objective and next sentence prediction task. The training data is sourced from a variety of sources, including books, articles, and web pages, and is cleaned and pre-processed to ensure high quality.

3. One of the key improvements of RoBERTa over BERT is the optimization of hyperparameters and training techniques. RoBERTa uses a larger batch size, more training data, and longer training time than BERT, which allows it to learn more robust and accurate representations of natural language. Additionally, RoBERTa uses dynamic masking during training, which randomly masks out different tokens in each training epoch, rather than using a static mask as in BERT. This encourages the model to learn more robust representations that are not biased towards specific patterns in the data.

4. We first import the required RoBERTa model and tokeniser.

5. Next we create a function to perform encoding of the input texts, here, the tweets. The function is called and we generate the id's and attention masks.

6. Followed by creation of RoBERTa model by first importing the model and then passing the input ids and attention masks generated by encoding.

7. Further on, we add the dense layer, dropout layer and final layer to the output generated.

8. This is followed by adding of Adam optimiser to the model for the optimization task.

9. We finally fit the data to the model and run the epoch cycles to complete the training.

10. The following is the model accuracy and model loss obtained:

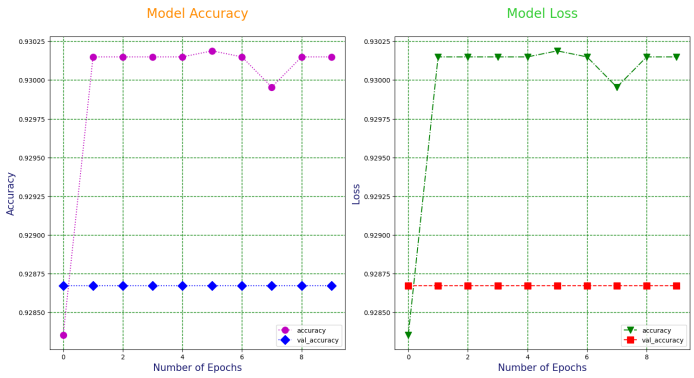

Figure 1: Model Accuracy and Model Loss

11. The ROC curve obtained is as shown below:

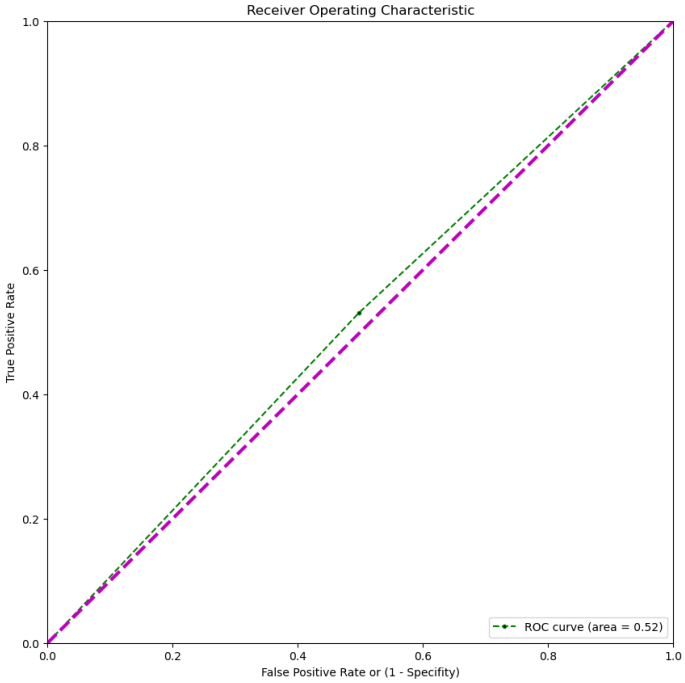

Figure 2: ROC Curve

5 Tokenisation & Vectorisation

1. Now before proceeding further we will perform certain operations to prepare the dataset for further steps.

2. First we split the dataset at hand in train and test data, in the ratio 80:20.

3. Next we tokenise and stemm the data, The purpose of tokenisation is to convert raw text into a format that is suitable for NLP tasks.Stemming, on the other hand, is the process of reducing words to their root form, called a stem. The purpose of stemming is to reduce the dimensionality of the text data and group together different forms of the same word.

4. Next we vectorise the data using TF-IDF (Term Frequency-Inverse Document Frequency), TF-IDF vectorization is a technique used to represent text data as numerical feature vectors. The purpose of TF-IDF vectorization is to transform raw text data into a format that is suitable for machine learning models and other data analysis tasks.

6 Oversampling using SMOTE

1. In an imbalanced dataset, the number of samples belonging to one class (in this case, non-hateful tweets) greatly outweighs the number of samples in the other class (in this case, (hateful tweets). This can lead to biased models that perform poorly on the minority class.

2. Synthetic Minority Over-sampling Technique (SMOTE) is a technique that creates synthetic examples of the minority class by creating new data points that are similar to the existing minority class samples, but with slight variations. Specifically, SMOTE works by selecting a minority class sample, finding its k nearest neighbors, and creating new synthetic samples along the line segments that connect the minority sample to its k nearest neighbors. By creating new synthetic samples, SMOTE increases the number of minority class samples, which can help balance the dataset and improve the performance of classification models on the minority class.

3. SMOTE can be used to create synthetic hateful tweets that are similar to existing ones, thereby increasing the number of hateful tweets in the dataset. This can help the classification model to better learn the patterns and characteristics of hateful tweets, and ultimately improve its accuracy in detecting hate speech.

6.1 Logistic Regression

1. We fit a Logistic Regression model on the balanced dataset.
2. The Evaluation Metrics obtained are as follows:

Table 1: Training and Validation Scores

Metric	Training Score	Validation Score
Accuracy	0.971	0.919
F1-Score	0.972	0.585

6.2 Naive Bayes Classifier

1. We fit a Naive Bayes Classifier on the balanced dataset.
2. The Evaluation Metrics obtained are as follows:

Table 2: Training and Validation Scores

Metric	Training Score	Validation Score
Accuracy	0.964	0.924
F1-Score	0.965	0.612

6.3 Random Forest Classifier

1. We fit a Random Forest Classifier on the balanced dataset.
2. The Evaluation Metrics obtained are as follows:

Table 3: Training and Validation Scores

Metric	Training Score	Validation Score
Accuracy	1.0	0.924
F1-Score	1.0	0.697

6.4 SVM

1. We fit a SVM model on the balanced dataset.
2. The Evaluation Metrics obtained are as follows:

Table 4: Training and Validation Scores

Metric	Training Score	Validation Score
Accuracy	0.987	0.919
F1-Score	0.987	0.563

7 Conclusion

1. The above techniques were applied to address the challenge of a highly imbalanced dataset, where the number of hateful tweets was significantly lower than the non-hateful tweets. This is a common problem in many natural language processing applications and can lead to biased models that perform poorly on real-world data.
2. Several techniques were employed to address this issue, including Random Forest Classifier, SMOTE technique, and manual tokenization using word_tokenizer and vectorization through TF-IDF vectorizer. We observed that Random Forest Classifier performed the best in both the training and validation datasets, indicating that it is a reliable and effective technique for this task.

3. Additionally, the usage of SMOTE technique helped to reduce the imbalance in the dataset and improved the performance of the models involved. This technique generated synthetic samples of the minority class, thereby making the dataset more balanced and reducing the chances of the model being biased towards the majority class.

4. Although RoBERTa had lower training accuracy than expected, while using layers defined by us, its validation accuracy was quite substantial at 93%, and the accuracy and loss remained almost constant with every epoch. This indicates that the model is able to generalize well to new, unseen data and can be relied upon for accurate predictions.

5. Furthermore, we found that manual tokenization using word_tokenizer and vectorization through TF-IDF vectorizer gave better results when combined with simpler classifiers than the inbuilt tokenization and vectorization of a complex model like RoBERTa. This highlights the importance of choosing the right combination of techniques that work well together for a given task.

6. Overall, we conclude that RFC with application SMOTE gave the best results, and the same can be used for labeling unseen tweets as either hateful or non-hateful. This approach has the potential to be used in real-world applications, such as automated content moderation on social media platforms, where accurate and unbiased classification of hateful content is critical.

8 Contribution

Every member equally contributed, we performed the task first individually then combined our findings to create the final project, oughly the work done was as follows:

1. G Mukund: Explored how different classifiers would work in the given scenario

2. Hrishikesh Makwana: Implemented the RoBERTa model and explored how it could be used in the project, resulting in the final model used by us.

3. Somshuvra Basu: Used SMOTE for Oversampling and explored how it could be best used for the oversampling step understanding the impact made on the findings of the classifiers used earlier.Also created the report in LaTeX.