SENTIMENT ANALYSIS OF TWEETS FOR HATE SPEECH DETECTION USING BINARY CLASSIFICATION ALGORITHMS AND BERT

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Manveer Kaur

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

November 2023

Fargo, North Dakota

North Dakota State University
Graduate School

Title

SENTIMENT ANALYSIS OF TWEETS FOR HATE SPEECH DETECTION USING BINARY CLASSIFICATION ALGORITHMS AND BERT

By

Manveer Kaur

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone A. Ludwig
Chair

Dr. Oksana Myronovych

Dr. Maria Alfonseca-Cubero

Approved:

November 15, 2023
Date

Dr. Simone A. Ludwig
Department Chair

ABSTRACT

In the modern world, social media wields a lot of power. Twitter, particularly, has provided people a platform to express their opinions about everything under the sun from mundane everyday life to politics, race, religion etc. It has often come under scrutiny for unabashed propagation of hate speech. This project employs natural language processing techniques on a corpus of tweets to detect hate speech. A total of 3538 unique tokens are identified that appear only in tweets classified as hate speech. With the help of data visualization techniques like word clouds and frequency distribution plots, it became evident that the occurrence of sexist, homophobic, and racist slurs is the most frequent in hate tweets. This implies that women, LGBTQ+ community, and people of color are the most targeted sections of society.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Simone A. Ludwig, my research advisor for providing valuable and timely guidance. I am also grateful to Dr. Oksana Myronovych and Dr. Maria Alfonseca-Cubero for their support and time to serve on my supervisory committee.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Artificial Intelligence (AI) is perhaps the most disruptive and important technology of our times. When OpenAI launched chatGPT in 2022, the focus of the entire world shifted to AI, machine learning and more specifically to natural language processing (NLP). NLP is a branch of machine learning that deals with unstructured data to analyze how elements of human language are structured together to impart meaning [1]. While the world is addressing important questions about restricting AI, so it does not replace humans, it is hard not to marvel at the kind of problems we are able to solve with NLP today. Some of the real-world applications of NLP are machine translation, text summarization, text classification, product recommendations, and sentiment analysis. Sentiment analysis specifically is an NLP technique that analyzes text to determine polarities (positive, negative, or neutral), emotions (happiness, sadness, or anger), or state of mind (interest or disinterest) towards target entities or topics. Despite the advances in machine learning algorithms and the advent of large language models, sentiment analysis is still restricted to solving business problems like gauging customer response from social media to certain products and marketing campaigns or devising action plans to market products based on customer feedback. The possibility of employing sentiment analysis for more humane tasks like identification of hate speech on social media and flagging harmful content remains seemingly untapped.

As of June 2023, 4.8 billion people use social media every day. An average user spends 864 hours a year scrolling through various apps. Social media algorithms are optimized for engagement which means they end up becoming echo chambers to cater to user's preferences. Twitter's ex CEO Jack Dorsey admitted how the algorithm enables the propagation of tweets

with negative sentiments over tweets with positive sentiments. With no accountability and unconditional freedom, the spread of negativity and hate on social media is rampant today.

United Nations has defined hate speech as “any kind of communication in speech, writing or behavior, that attacks or uses pejorative or discriminatory language with reference to a person or a group on the basis of who they are, in other words, based on their religion, ethnicity, nationality, race, color, descent, gender or other identity factor”. Usage of hate speech often comes in contention with freedom of speech. But given the impact hate speech can have, it becomes important to flag inflammatory posts on social media for removal. However, this task cannot be wholly accomplished by targeting the usage of certain words identified in the hate speech lexicon. Understanding the nuances and context of certain words in a sentence is important for hate speech detection. Not every offensive text can be classified as hate speech. For instance, the usage of the ‘n’ word in African American rap music can be deemed offensive but not hate speech [2]. Besides, social media language comprises of made-up words, broken vocabulary, incorrect usage of grammar. In addition, it consists of abbreviations, nonstandard punctuation, improper spelling, emoticons, and slangs. Context-aware ways to utilize ambiguity are either nonexistent or inefficient due to the lack of facial expressions, visual, and tone-of-voice clues.

Despite the advances in the field of NLP, it is still a difficult task to deduce the underlying meaning of a sentence. The focus of this study is to use sentiment analysis techniques to extract the sentiment of text. This will be incorporated with various data processing techniques aimed at creating a clear distinction between hate speech and generally offensive speech to understand the nuances. This can be helpful in ensuring social media platforms remain clean and wield less destructive power.

In this project, sentiment analysis techniques are used along with data visualization techniques aiming to:

(a) Identify and flag tweets that violate hate speech policies.

(b) Identify the sections of society that are most vulnerable and susceptible to hate speech attacks.

2. LITERATURE REVIEW

Hate speech depends greatly on nuance. There is a fine line between hate speech and offensive language. This makes the task of sentiment analysis for hate speech detection hard. Separating the two is crucial in hate speech detection. In the recent times, a lot of studies have been conducted on automatic detection of hate speech. Lexical detection methods classify all texts containing certain terms as hate speech, and therefore, tend to have low precision [2]. Kwok and Wang in their study on detecting tweets against Black population found that 86% of the time a tweet was classified as racist was because of the use of offensive words. The usage of anti-black racist words is quite prevalent among the black population and the majority of the time the usage can be termed offensive rather than hate speech or racist speech. Similarly, the word 'gay' can be used in contexts both related and unrelated to hate speech [3]. This once again underlines the importance of nuance in hate speech detection.

Machine learning algorithms based on feature engineering are widely used in the field of hate speech detection [4]. Gitari et al. (2015) took a lexicon-based approach and designed various sentiment features for hate speech detection that accounted for sentence structure [5]. For instance, the occurrence of a relevant noun (like 'Blacks', 'Jews') and verb (like 'kill', 'loot', 'beat') in a sentence. Silva et al. in their study to analyze the targets of hate on social media used a strategy that searches for sentence structures *I <intensity> <user intent> <hate target>* [6]. This template captures when hate is targeted towards a group of people (e.g., 'I just hate Jews'). Bag-of-words approaches lead to high false positives as the presence of offensive words leads to misclassification [3]. Burnap et al. derived classification features from content of each tweet, grammatical dependencies between words, incitement to respond with antagonistic actions to

create a supervised machine learning classifier [7]. However, it just conflated offensive language with hate speech making it difficult to identify hate speech.

Deep learning-based methods have shown significant promise. Zhang et al. used a Convolutional Neural Network (CNN) and a Gated Recurrent Unit (GRU) to learn higher-level features [8]. Tekiroglu et al. constructed a dataset based on hate speech and its responses and used the pre-trained language model GPT-2, for hate speech detection [9].

3. METHODOLOGY

Opinion mining is a subfield of linguistics and natural language processing that deals with sentiment analysis [10]. It evaluates the degree of polarity of words and sentences and extracts views and feelings from textual data [11]. Sentiment analysis is an excellent way to know how the target audience or consumers feel about a particular product, campaign, or even an idea. The origin of sentiment analysis can be traced back to 1950s, when it was primarily used on written paper documents. In the modern world, sentiment analysis is used in every field imaginable. Governments use sentiment analysis to sense public mood on policy announcements, political parties use it to predict public response to political campaigns, large corporate houses use sentiment analysis to predict the consumer response to their products or marketing campaigns or to see how their competitor's products are received by consumers.

For this project, two different approaches have been used for performing sentiment analysis on tweets for hate speech detection. The first approach involved building three different binary classification models – multinomial naïve Bayes classifier, random forest classifier, and logistic classifier to study their performance. The reason these specific algorithms were chosen was because of the success they have showed in previous work in this field. Figure 1 shows the schematic process flow diagram to building these models. The initial steps are the same for all three classifiers. We collect data either by scraping Twitter or from public datasets. Data is then preprocessed, and a vocabulary of tokens is built. Data visualization techniques help in understanding our data better and generalizing about the dataset. The textual tokens are then converted into numbered form or vectors to be fed to our machine learning models which perform the task of classification.

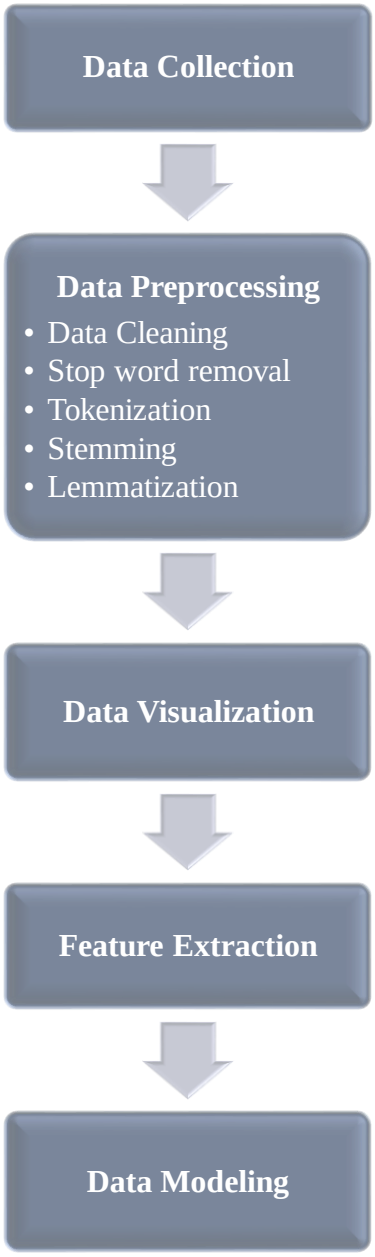

Figure 1. Schematic process-flow diagram for binary classification algorithms

The second approach involves fine-tuning a pre-trained language model Bidirectional Encoder Representations from Transformers (BERT) for sequence classification. In a transformer the input consists of sentence pairs. The self-attention mechanism in the transformer allows BERT to model many downstream tasks, whether they involve single text or text pairs

[13]. For each downstream task, the inputs and outputs are plugged into BERT and all parameters are fine-tuned end-to-end.

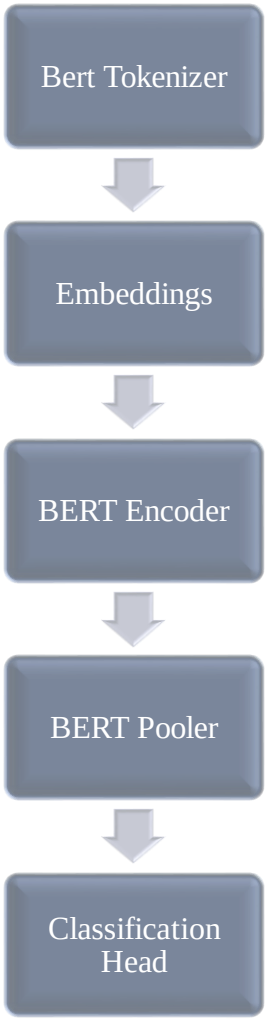

Figure 2. Architecture of a BERT model

For text classification tasks the input representation is a single sentence instead of a pair of sentences (e.g., <Question, Answer>) because the second sentence is relevant only in next-sentence prediction tasks and not in text classification tasks. Figure 2 shows the architecture of a BERT model. The encodings from a BERT tokenizer are passed into the Embeddings module. In simple terms, embeddings are vector representations of tokens. Then, embeddings are passed into the encoder module. The base BERT model has 12 transformer layers in the encoder. Output

from each layer feeds into the other in a sequential manner. The BERT pooler returns an embedding for the [CLS] token which is passed on to the classification head, which in turn returns the classification for the text sequence.

3.1. Preliminaries

3.1.1. Python Libraries

The code for this project was written in Python and executed in Jupyter notebooks. The various Python libraries used to accomplish different tasks in this project are listed below:

- Pandas, Numpy – are used for data loading and manipulation
- Regular Expression – was used for data cleaning purposes
- NLTK – was used for data preprocessing tasks like stop word removal, tokenization, stemming, lemmatization.
- Scikit-learn – was used for feature extraction with TF-IDF and CountVectorizer. It was also used for data modeling.
- Seaborn, Matplotlib, Yellowbrick, WordCloud – were used for data visualization tasks.
- Transformers, PyTorch – were used for implementing the transformer model.

3.1.2. Environment

All the experiments and computations were performed on a personal Mac computer with M1 chip and 8 GB of RAM.

3.2. Data Collection

The data for this project is sourced from Cornell University's 2017 research on Automated Hate Speech Detection and the Problem of Offensive Language [12]. The corpus contains a random sample of about 25,000 tweets that are manually labeled into three categories:

hate speech, offensive language, and neutral as shown in Figure 3. The data, though reliable, is highly imbalanced and contains merely 5% tweets that are labeled as hate speech. Association of Computational Linguistics (ACL) provides labeled data with tweet IDs of tweets containing hate speech. In the past, the Tweeter API could be used to scrape data from twitter using the tweet IDs [15]. But, with the recent changes at X, scraping has been rendered almost impossible. A publicly available corpus of hate tweets extracted from the ACL dataset was then used to balance the dataset. The new balanced dataset is shown in Figure 4. Initial exploratory data analysis made it evident that the new dataset had a total of 8,337 tweets with 4,174 tweets classified as hate speech with label '1' and 4,163 tweets classified as neutral speech with label '0'. This dataset had an equitable class distribution of tweets as depicted by Figure 5 and 6.

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	3	0	2	1	1	!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	4	6	0	6	0	1	!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
5	5	3	1	2	0	1	!!!!!!!!!!!!!!!"@T_Madison_x: The shit just...
6	6	3	0	3	0	1	!!!!!"@__BrighterDays: I can not just sit up ...
7	7	3	0	3	0	1	!!!!“@selfiequeenbri: cause I'm tired of...
8	8	3	0	3	0	1	" & you might not get ya bitch back & ...
9	9	3	1	2	0	1	" @rhythmixx_ :hobbies include: fighting Maria...

Figure 3. Manually labeled dataset from Cornell study

	Unnamed: 0	text	class
0	0	Drasko they didn't cook half a bird you idiot ...	1
1	1	Hopefully someone cooks Drasko in the next ep ...	1
2	2	of course you were born in serbia...you're as ...	1
3	3	These girls are the equivalent of the irritati...	1
4	4	RT @YesYoureRacist: At least you're only a tin...	1
5	5	@MisfitInChains @oldgfatherclock @venereverita...	1
6	6	RT @Dreamdefenders: Eric Holder from #ferguson...	1
7	7	RT @AntonioFrench: I spent the morning at the ...	1
8	8	RT @harikondabolu: In his #Ferguson testimony,...	1
9	9	"He can't be a server at our restaurant, that ...	1

Figure 4. Data obtained from publicly available ACL dataset

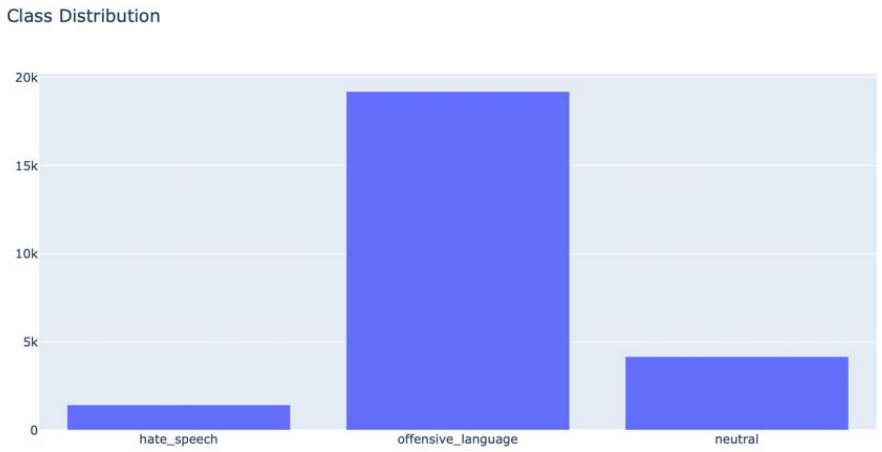

Figure 5. Class distribution of tweets before balancing dataset

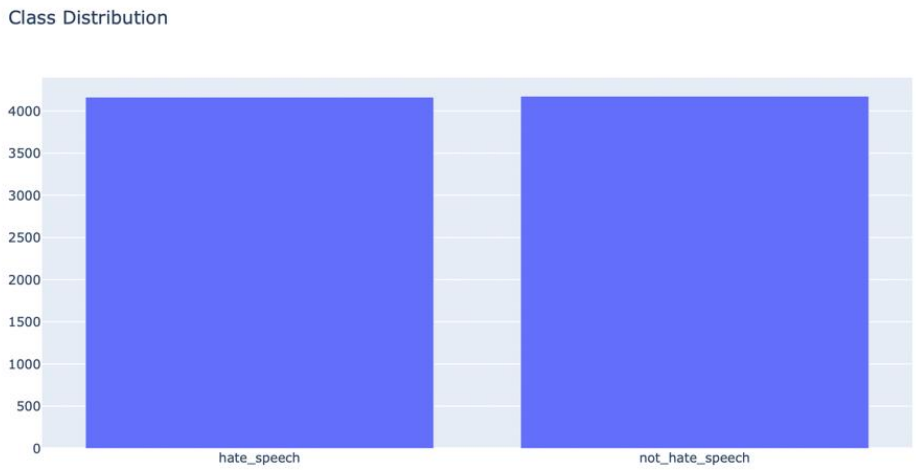

Figure 6. Class distribution of tweets after balancing dataset

3.3. Data Preprocessing

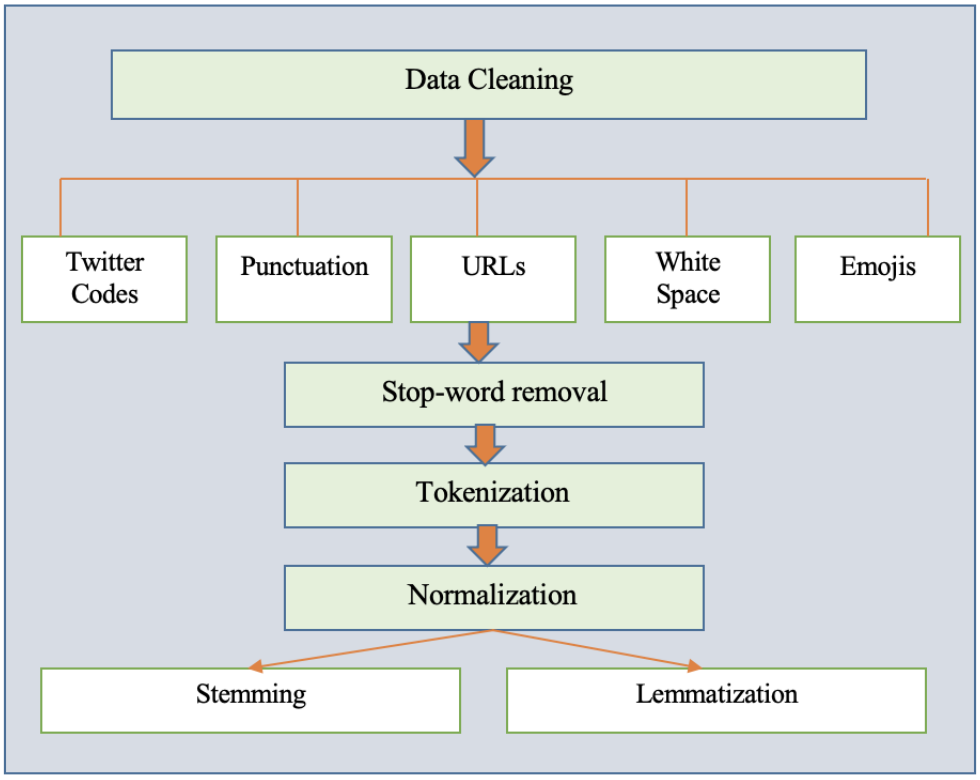

Figure 7. Data Preprocessing Tasks

3.3.1. Data Cleaning

For building a classification model, data cleaning is one of the most crucial steps. As shown in Figure 7, it is the first step in data preprocessing. Removal of unwanted columns is the foremost step. In this dataset, only two columns hold significance i.e., tweet text and the classifier (1 for hate speech and 0 for neutral speech). The second step in data cleaning is preprocessing tweets. The raw tweet text comprises of hashtags, twitter codes of retweet and quote tweet, punctuation, emojis, unicode characters, whitespaces, URLs. Python's regular expression module is used to remove all these characters from tweets. Figures 8 and 9 show the tweets before and after cleaning, respectively.

```
0        !!! RT @mayasolovely: As a woman you shouldn't...
1        !!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2        !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3        !!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4        !!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
                                ...
24778    you's a muthaf***in lie &#8220;@LifeAsKing: @2...
24779    you've gone and broke the wrong heart baby, an...
24780    young buck wanna eat!!.. dat nigguh like I ain...
24781                  youu got wild bitches tellin you lies
24782    ~~Ruffled | Ntac Eileen Dahlia - Beautiful col...
Name: tweet, Length: 24783, dtype: object
```

Figure 8. Tweets before performing data cleaning steps

```
0        as a woman you shouldnt complain about cleanin...
1        boy dats coldtyga dwn bad for cuffin dat hoe i...
2        dawg you ever fuck a bitch and she start to cr...
3                                 she look like a tranny
4        the shit you hear about me might be true or it...
                                ...
24778    yous a muthafin lie right his tl is trash now ...
24779    youve gone and broke the wrong heart baby and ...
24780    young buck wanna eat dat nigguh like i aint fu...
24781                  youu got wild bitches tellin you lies
24782    ruffled ntac eileen dahlia beautiful color com...
Name: tweet, Length: 24783, dtype: object
```

Figure 9. Tweets after performing data cleaning steps

3.3.2. Stop Word Removal

Stop words are the most commonly occurring words in a sentence, comprising of articles, prepositions, conjunctions, pronouns, etc. and do not add much information to text. Python's NLTK library provides a corpus of words that are considered stop words. Using this corpus, the cleaned tweets are processed such that all the stop words are excluded from the tweets.

3.3.3. Tokenization

Tokenizers divide strings into lists of substrings or sentences into a list of individual words. The NLTK word tokenizer is used to achieve tokenization.

3.3.4. Normalization

3.3.4.1. Stemming

Stemming is the process of reducing inflected or sometimes derived words to their word stem, base or root form. For example, the words 'run', 'running', and 'runner' will be stemmed to their root form 'run'. The NLTK library provides various stemmers. For this project, Porter Stemmer is used to remove morphological affixes from words.

3.3.4.2. Lemmatization

Lemmatization is the process of reducing a word to its lemma by using morphological analysis of the words using dictionaries. In essence, the stem of a word may or may not be a meaningful word, but lemma is always a meaningful word.

3.4. Data Visualization

Once the data is preprocessed by doing stop word removal and tokenization, frequency distribution plots and word clouds of words within the whole corpus are generated. A frequency distribution tells us the frequency of each token within and across the corpus. CountVectorizer is used to transform the tokens into vectors based on the frequency of each token in the corpus. It

creates a matrix in which each token is represented by a column and each text from corpus is represented by a row in the matrix. The value of each cell is the count of a token in the particular text. The count vectorized matrix is then used to plot the frequency distribution plots. These plots and word clouds of most used tokens helped in making important deductions about data. The total number of tokens in tweets labeled as hate speech were 26,101. The number of unique hate tokens were 6,378. Similarly, the number of tokens generated from tweets labeled neutral were 26,012 with 8,290 unique tokens. The word clouds in Figures 11 & 13 and frequency distribution plots in Figures 10 & 12 inform us about the most frequently used words in both hate tweets and neutral tweets.

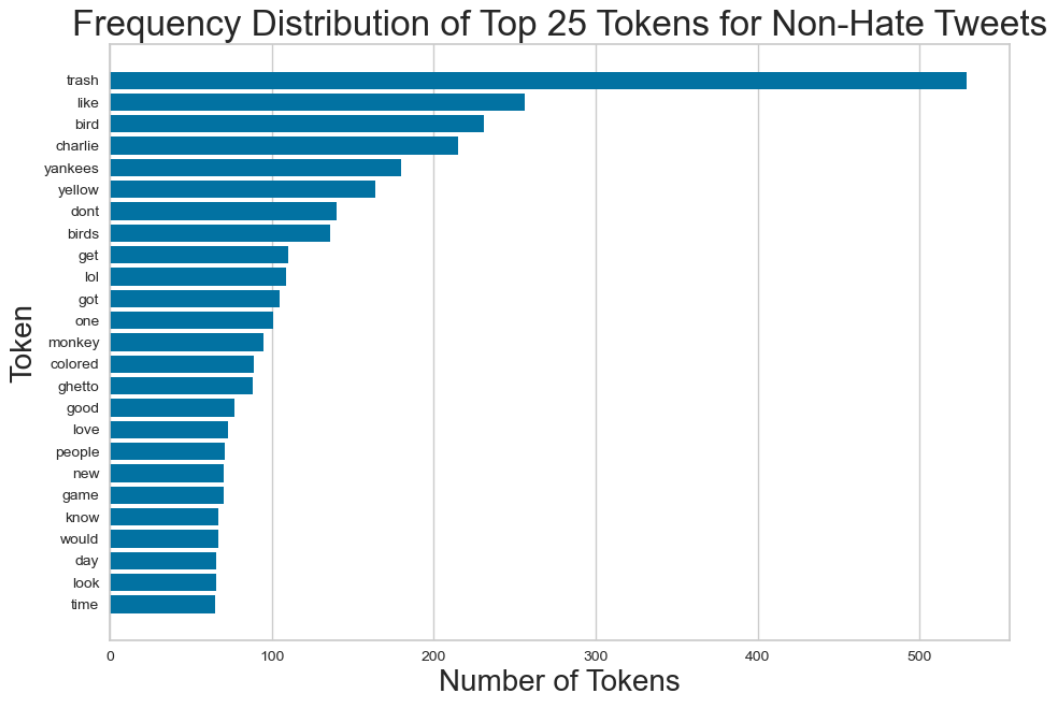

Figure 10. Frequency distribution for top 25 tokens in tweets that were not hate speech

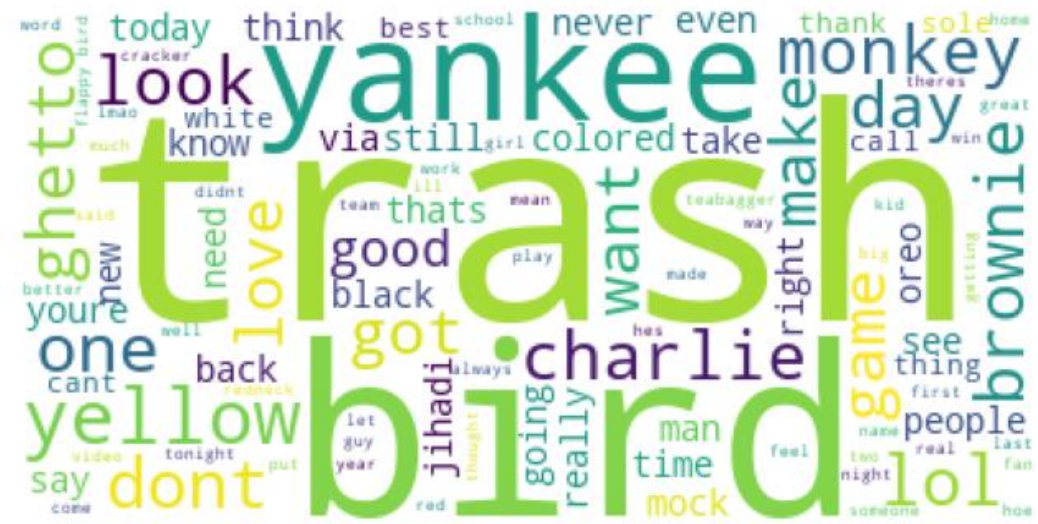

Figure 11. Word cloud representing top 100 tokens in tweets that were not hate speech

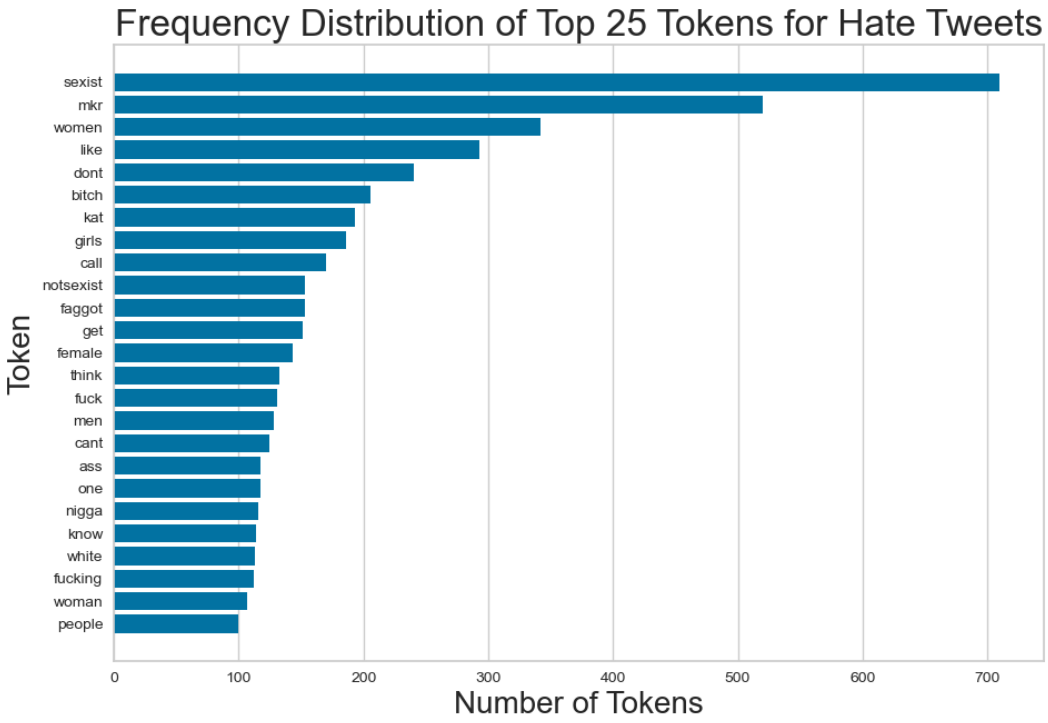

Figure 12. Frequency distribution for top 25 tokens in tweets that were hate speech

Figure 13. Word cloud representing top 100 tokens in tweets that were hate speech

From the data visualizations, it became evident that both set of tweets had some common tokens. This made it difficult for machine learning models to make correct predictions. After taking the set difference of the two sets of tokens, 3,538 unique tokens were extracted that appeared only in hate tweets. This helped in reducing false negatives.

3.5. Feature Extraction

Raw text cannot be directly fed into machine learning models. Feature extraction is the process of converting raw text into a set of numerical or categorical features. These numerical vectors act as input for machine learning models which then analyze and classify them. There are multiple feature extraction techniques. For this project, TF-IDF is used.

3.5.1. Term Frequency – Inverse Document Frequency (TF/IDF)

TF-IDF is a measure of importance of a word in a document, adjusted for the fact that some words appear more frequently than others. TF-IDF works by determining relative frequency of words in a specific document compared to the inverse proportion of that word over

the entire document corpus [16]. Essentially, it is a product of two statistics, term frequency and inverse document frequency.

Term Frequency (TF) is the relative frequency of term *t* in document *d*. Logarithmically scaled TF is formulated as:

$$tf(t,d) = \log(1 + f_{t,d})$$

where, $f_{t,d}$ is the frequency of term *t* in *d.*

Inverse Document Frequency (IDF) is a measure of how much information the word provides. It is logarithmically scaled inverse fraction of documents that contain the word.

$$idf(t,D) = \log\frac{N}{\{d \in D: t \in d\}}$$

where, $f_{t,d}$ is the frequency of term *t* in *d.*

N is the total number of documents in the corpus, N = |D|

$\{d \in D: t \in d\}$ is the number of documents where the term t appears

The TF-IDF is then calculated as a product of TF and IDF.

$$tfidf(t,d,D) = tf(t,d) * idf(t,D)$$

4. DATA MODELING

4.1. Data Modeling with Binary Classification Algorithms

Binary classification is the task of classifying the elements of set into two distinct classes based on a classification rule. Sentiment analysis of tweets is a task of classifying a set of tweets into two categories: 'hate speech' and 'neutral speech'. This is done by implementing supervised machine learning algorithms. Supervised machine learning is a paradigm where we train the model on labeled data and then test it to predict or classify unlabeled data. I have employed three binary classification algorithms models: multinomial naïve Bayes, random forest classifier, and logistic regression. The reason for choosing these algorithms is their popularity and success in text classification tasks. In addition to using the traditional classification algorithms, I have also fine-tuned a pre-trained transformer model BERT.

4.1.1. Multinomial Naïve Bayes

Naïve Bayes is a learning algorithm that is frequently employed to tackle text classification problems [17]. Multinomial naïve Bayes (MNB) is widely used for assigning documents to classes based on statistical analysis of their contents. It does so by determining the probability that a document (fragment of text) belongs to a particular class. Given the samples *S* intended to be classified, each sample in *S* is defined as a string that occurred in one or multiple documents from a class *C*. To perform the classification, the terms in *S* are represented by a vector *W*. Each feature w_i of *W* is an occurrence frequency of the corresponding *i*-th term in the documents from class *C*. $p(w_i \mid C)$ is the probability that the terms in *S* occurred in documents from class *C*. The Bayesian probability is computed as follows:

$$p(C_k|W) = \frac{p(C_k) * p(W|C_k)}{p(W)}$$

The main idea behind naïve Bayes is that all features in W independently contribute to the probability that *S* belongs to C_k. MNB assumes that *S* is represented by feature vectors *W*. Each feature vector w_i is the count with which the *i*-th term from *S* occurred in the text already classified to class *C* [18].

4.1.2. Random Forest Classifier

Random forest classifier is a classification algorithm that consists of multiple decision trees forming an ensemble. It is an extension of bagging method and utilizes randomness in creating decision trees. Each tree comprises of data sample drawn from training set with replacement called the bootstrap sample. Classification is done by aggregating the predictions of individual decision trees to identify the most popular result [19].

4.1.3. Logistic Regression

Logistic Regression is a Machine Learning method that is used to solve classification issues. It is a statistical model that is based on the probability of an event taking place. The logistic regression hypothesis suggests that the cost function be limited to a value between 0 and 1 [21]. A Logistic Regression model utilizes a sophisticated cost function known as the 'Sigmoid function', which is calculated for any value x as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Logistic Regression utilizes the gradient descent algorithm to find the optimum weights and biases for the model to minimize cost.

4.2. Data Modeling with BERT

Bidirectional Encoder Representations from Transformers (BERT) is a large-scale transformer-based language model that can be finetuned for a variety of tasks. The first

transformer model was introduced in the very popular ‘Attention is all you need’ paper [22]. It comprises of encoder-decoder architecture based on attention layers. Unlike traditional Recurrent Neural Networks (RNN) where input sequence is fed in a continuous method, one word at a time to generate word embeddings, a transformer allows passing all the words simultaneously and generating word embeddings simultaneously as well.

In a transformer, the encoder block takes in input embeddings along with positional encodings for each word in a sequence. It comprises of a multi-head attention block and a feed-forward neural network, where it applies self-attention to generate attention vectors for all words parallelly. The decoder also operates in a similar fashion. In addition to the multi-head attention block and feed forward network, it has masked attention block and a linear and softmax layer. It takes in output encodings along with the representations produced by encoder to generate one word at a time from left to right. Therefore, the transformer is comprised of two separate blocks or mechanisms – an encoder that reads raw text input and the decoder which produces a prediction for the task.

BERT uses transformer at its core but since its goal is to generate a language model, only the encoder mechanism is employed [23]. BERT framework consists of two steps: pre-training and fine-tuning. The model is pre-trained on unlabeled data over different tasks. For fine-tuning, the model is first initialized over pre-trained parameters and then all the parameters are fine-tuned with labeled data for downstream tasks [23]. A pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for wide range of downstream tasks such as question answering and language inference, without substantial task-specific architecture modifications [24].

The BERT model architecture for sequence classification as shown in Figure 3 comprises of the tokenizer, embeddings module, encoder, pooler, and a classification head.

4.2.1. BERT Tokenizer

BERT tokenizer uses sub-word tokenization algorithm as shown in Figure 14, which tends to strike a balance between sequence length and vocabulary size. The algorithm builds a vocabulary of tokens of a defined size by selecting and merging the highest scoring character pairs in the text dataset. Each token in the vocabulary is assigned an integer ID and embeddings of each sentence are generated using these IDs in one-hot encoding fashion.

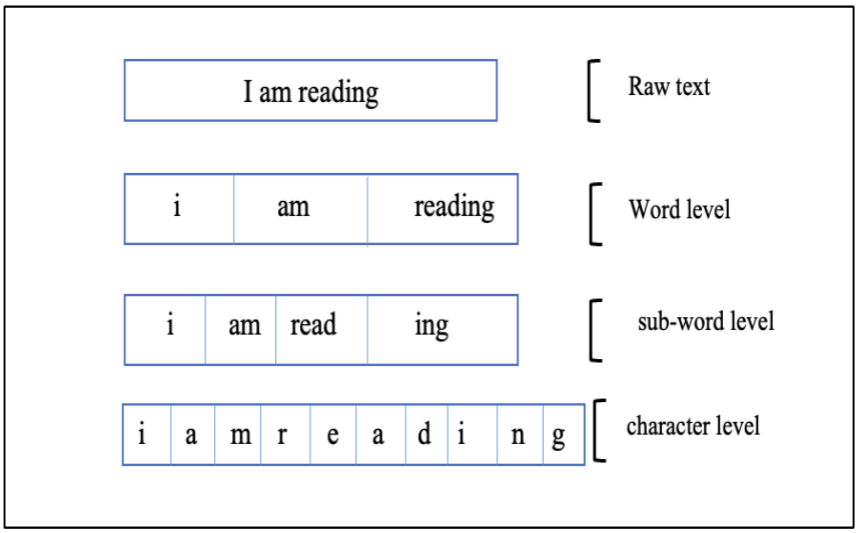

Figure 14. Tokenization of a sequence at a word, sub-word and character level

The BERT tokenizer generates an output of processed text which has three special tokens:

- Classification token: The [CLS] token signifies the beginning of a sentence.
- Separation token: The [SEP] token is placed at the end of each sentence in a text sequence. This is how BERT differentiates between sentences.
- Padding token: The [PAD] token is placed at the end of short sentences as many times as needed to ensure that all sentences are of the same length.

Apart from the special tokens, output contains the following three types of integer encodings:

- Input IDs: There are 30522 tokens in the defined vocabulary of BERT base model. Each token maps to a unique integer ID. These IDs in a sequence form the input IDs.
- Attention mask: Attention masks does the task of assigning the value 0 to padding tokens and 1 to non-padding tokens.
- Token type IDs: Token type IDs are all 0 for text classification tasks and have no significance. They are used in sentence prediction tasks.

4.2.2. Embedding Module

Embeddings are the vector representations of tokens. The encodings from the tokenizer and fed into the encoder which in turn generates the following three embeddings for each token:

- Word embeddings: Each input ID is replaced with corresponding row vector from word embedding lookup table which has one row dedicated to each unique token in the vocabulary (30522 rows).
- Token type embeddings: These again have no significance for sequence classification tasks but otherwise signify which sentence from input sequence the token is from when there are two sentences.
- Positional embeddings: Positional embeddings let the model know about the order of tokens in a sentence. The maximum sequence length in base BERT model is 512, the corresponding positional embeddings will be vector of consecutive integers from 0 to 512.

All three embeddings are added to generate the final embeddings of the tokens.

4.2.3. Encoder

The encoder in BERT base model has 12 identical layers. Output from one layer is fed into the next layer sequentially. The two main components of encoder layers are:

- Multi-head self-attention: The self-attention mechanism allows the model to consider the context of a word in a sentence. It weighs the importance of the token by comparing it with other tokens in that sentence. The embedding vector is run through a linear layer three times and outputs three vectors of same dimension called query, key, and value vectors. Dot product of query and key vectors is then computed to measure the similarity between the vectors. Larger the scalar output of the dot product, the higher the similarity between two vectors. The scaled similarity score is then passed through a softmax function before being matrix multiplied with the value vector, resulting in enriched embeddings. The enriched embeddings inform us to what degree a token is paying attention to other token vectors. The 12 transformer layers with multiple attention heads allow the model to capture different kind of relationships between tokens and generating richer token representations.
- Position-wise feed-forward network: The feed-forward network aims to improve the model capacity by continually enriching token representations. It projects the inputs to higher embedding size and introduces a non-linearity through GeLU activation function. The GeLU function helps to capture non-linear relationships between tokens.

4.2.4. Pooler

The pooler takes the output from encoder, which is a matrix and applies transformations on it to essentially condense it into a single vector representation. The transformations involve

processing the classification token [CLS] through a linear projection (dense layer) and a non-linear activation function.

4.2.5. Classification Head

The output from the pooler is passed through the classification head, which projects the pooled embedding into a space with dimensionality equal to the number of different classes. The outputs The outputs are the logits for which there is one value for each class. Taking the maximum value of these logits will give us the predicted class.

5. RESULTS AND OBSERVATIONS

5.1. Model Performances

The metrics used to analyze the performance of the models are F1score, recall, and accuracy. F1 score is used because the initial data was imbalanced. It provided a more accurate measure of the model's performance over other metrics. Recall is used because it is the measure of the proportion of hate tweets correctly identified. Accuracy is the measure of how many correct predictions the model made, it determines the overall correctness of the model. However, correctly classifying hate tweets is more important than misclassifying non-hate tweets, therefore, recall is preferred over accuracy. Also, since F1score is the weighted average of accuracy and recall, overall, it gives a better idea about the models than both accuracy and recall. The metrics were computed from the confusion matrix using Equation (1) through (3).

$$F1\ Score = \frac{2TP}{2TP+FP+FN} \tag{1}$$

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

where:

TP = number of positive samples correctly classified

TN = number of negative samples correctly classified

FN = number of positive samples incorrectly classified

FP = number of negative samples incorrectly classified

The original dataset was highly imbalanced with only about 6% tweets classified as hate speech. The model performance results for the imbalanced dataset are summarized in Table 1. The three binary classification algorithms performed poorly with logistic regression leading with an F1 score of 18.49%. Multinomial naïve Bayes classifier failed to make even a single hate

speech classification. It classified all the tweets in the validation set as neutral. BERT model performed much better than the other algorithms, it attained an F1 score of 47.16% after training for over 40 hours. The results offered an interesting insight that accuracy is not a great metric for measuring imbalanced data. With almost all tweets classified as neutral, the accuracy of all three models was over 94%.

Table 1. Performance of classification algorithms on unbalanced dataset

Algorithm	F1 Score	Recall	Accuracy
Multinomial Naïve Bayes	0%	0%	94.50%
Random Forest Classifier	17.93%	11.03%	94.09%
Logistic Regression	18.49%	11.38%	94.13%
BERT	47.16%	46.50%	94%

Once the data was balanced, the algorithms showed exponential improvement in performance. Of the three classification algorithms, random forest classifier had an edge with an F1 score of 91.32%. Table 2 shows the performance metrics of the three binary classification algorithms as well as the transformer model. The BERT model for sequence classification was trained over 3 epochs. The training time was a little over 16 hours. But the transformer model outperformed all the three classification algorithms with an F1 score of 92.26%.

Table 2. Performance of different algorithms on balanced dataset

Algorithm	F-1 Score	Recall	Accuracy
Multinomial Naïve Bayes	89.43%	91.85%	89.41%
Random Forest Classifier	91.32%	93.40%	91.12%
Logistic Regression	90.35%	86.45%	90.76%
BERT	92.26%	91.13%	93.16%

Table 3. Performance of BERT model over 3 epochs

Epoch	F-1 Score	Training loss	Validation loss
Epoch 1	90.45%	0.42450	0.24286
Epoch 2	91.78%	0.21206	0.21119
Epoch 3	92.26%	0.16194	0.21061

The BERT model attained very good performance after 2 epochs as shown in Table 3. The training error reduced dramatically from 0.42 to 0.21 within 2 epochs.

5.2. Confusion Matrices

A confusion matrix summarizes the performance of a classification model. Figures 15 through 18 show the confusion matrix from multinomial naïve Bayes, random forest classifier, logistic regression, and the BERT model.

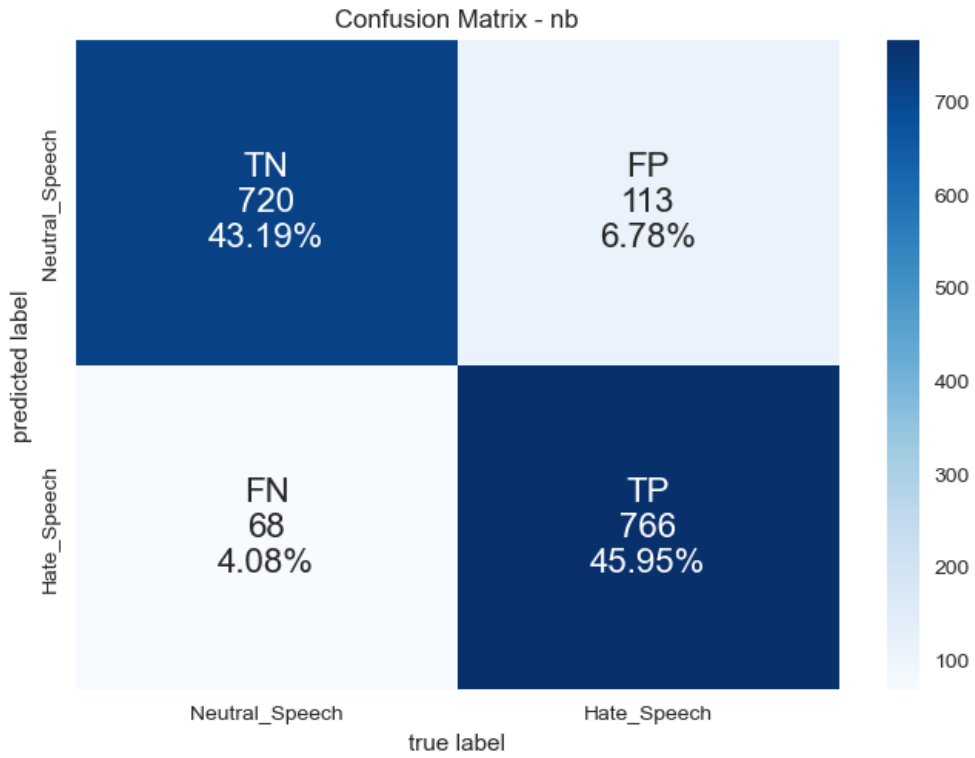

Figure 15. Confusion matrix for Multinomial Naïve-Bayes algorithm

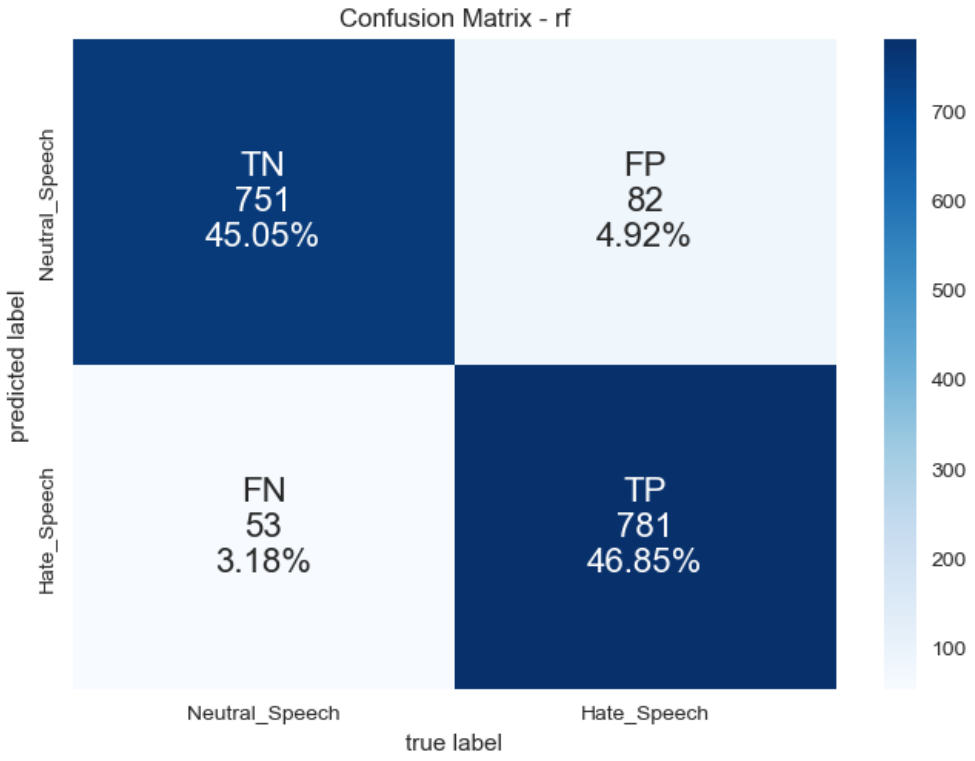

Figure 16. Confusion matrix for Random Forest classifier algorithm

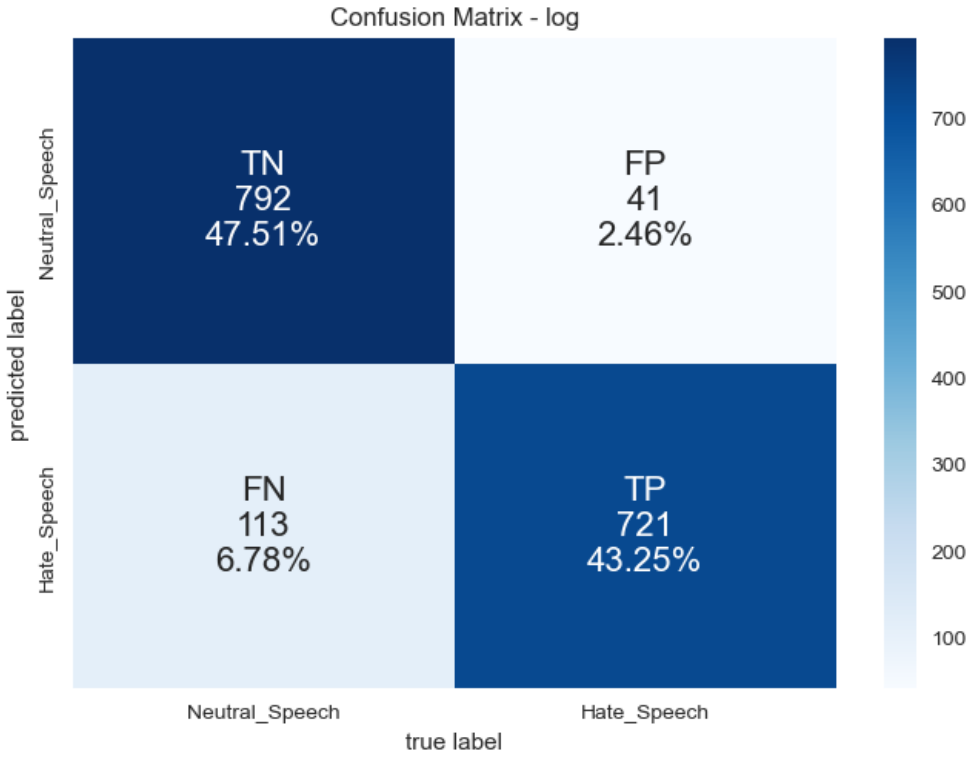

Figure 17. Confusion matrix for Logistic regression algorithm

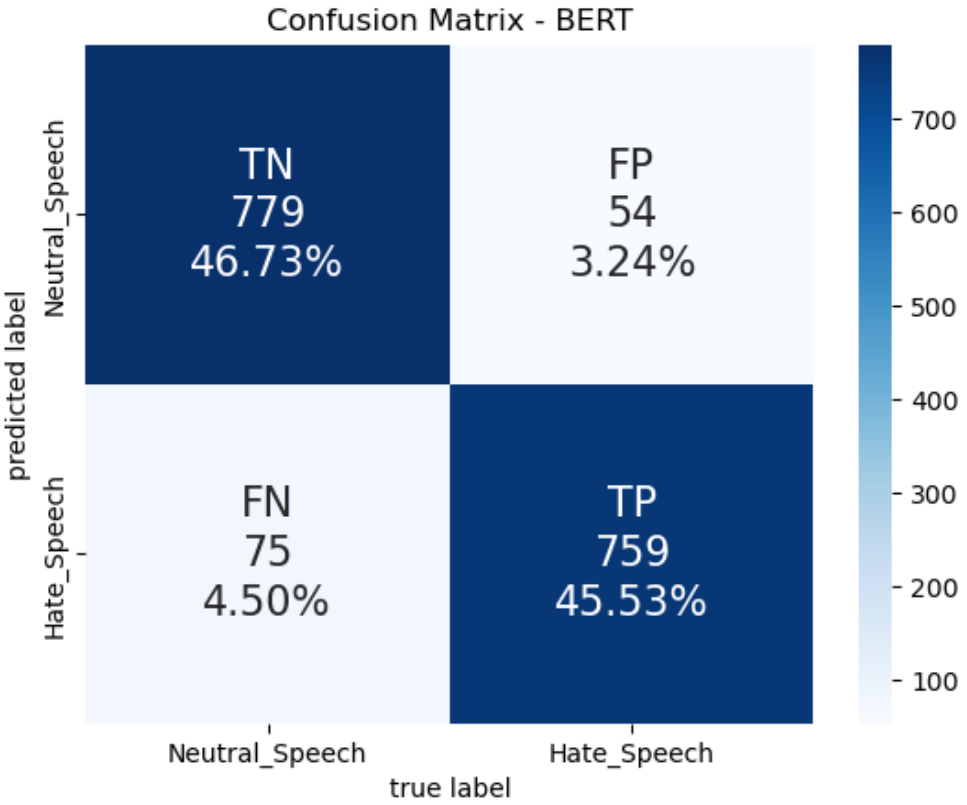

Figure 18. Confusion matrix for BERT model

5.3. Findings

The data visualizations pointed out very interesting data trends. On generating frequency distribution plots (Figure 19) and word cloud (Figure 20) of 3,538 words that were unique to hate tweets, it became evident that the majority of hate speech was targeted at women (with words like ‘sexist’, ‘feminist’, ‘bi**hes’, ‘sexism’, ‘women against feminism’, etc.), the LGBTQ+ community (with the ‘f’ word), and the people of African descent (with the different variations of ‘n’ word).

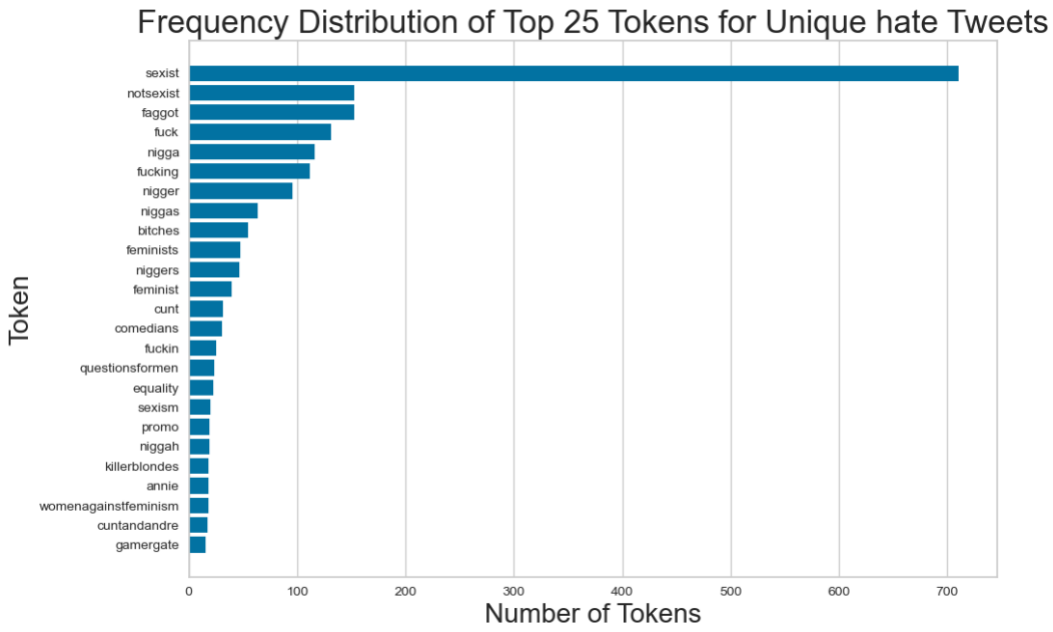

Figure 19. Frequency distribution for top 25 tokens in tweets that were unique to hate tweets

Figure 20. Word cloud representing top 100 tokens in tweets that were unique to hate tweets

The observations help us have a better understanding of the online discourse. It helps us to draw dark but very clear conclusions about us as a society. But it also forces us to have more discussions on the persistent persecution of certain sections of society on online forums. More importantly, it provides us a way forward to tackle hate speech on social media.

6. CONCLUSION AND FUTURE WORK

Machine translation is the not the next ‘big thing’, it is the current ‘big thing’. Despite the underlying complications presented by nuance and contextual meaning of words, there has been tremendous progress in this field in a very short amount of time. This project has been able to demonstrate the capability of classification algorithms in identifying hate speech in text. Transformer based models are, without any doubt, one of the strongest contenders in the field of Natural Language Processing. However, it is difficult to pick one particular machine learning classifier as optimum model for opinion mining because a machine learning model is as good as the data it is trained on. This project is limited by the availability of data. Realtime data scraping from Twitter by using the keywords identified in this project could help provide even better data insights and building a better model. Data sources can also be diversified by collecting data from different social media platforms. Additionally, with new advancements happening in the field of machine translation, the capabilities of sentiment analysis can also be extended to provide multi-lingual support as huge amount of the data from non-English speaking countries on social media is generally composed of words from multiple languages.

REFERENCES

[1] S. Bird, E. Klein, and E. Loper, 'Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit', O'Reilly, 2009.

[2] Y. Wang, J. Guo, C. Yuan, and B. Li, 'Sentiment Analysis of Twitter Data', Applied Sciences, vol. 12, p. 11775, 2022.

[3] I. Kwok and Y. Wang, 'Locate the Hate: Detecting Tweets against Blacks', Proceedings of the AAAI Conference on Artificial Intelligence, vol. 27, pp. 1621-1622, 2013.

[4] X. Zhou, Y. Yong, X. Fan, G. Ren, Y. Song, Y. Diao, L. Yang, and H. Lin, 'Hate Speech Detection Based on Sentiment Knowledge Sharing', Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, pp. 7158-7166, 2021.

[5] D. Njagi, Z. Zuping, D. Hanyurwimfura, and J. Long, 'A Lexicon-based Approach for Hate Speech Detection', International Journal of Multimedia and Ubiquitous Engineering, vol. 10, pp. 215-230, 2015.

[6] L. Silva, M. Mondal, D. Correa, F. Benevenuto, and I. Weber, 'Analyzing the Targets of Hate in Online Social Media', Proceedings of the International AAAI Conference on Web and Social Media, vol. 10, pp. 687-690, 2016.

[7] P. Burnap and M. Williams, 'Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decision Making: Machine Classification of Cyber Hate Speech', Policy & Internet, vol. 7, pp. 223-242, 2015.

[8] Z. Zhang, D. Robinson, and J. Tepper, 'Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network', The Semantic Web, vol. 10843, pp. 745-760, 2018.

[9] S. S. Tekiroglu, Y.-L. Chung, and M. Guerini, ‘Generating Counter Narratives against Online Hate Speech: Data and Strategies’, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 1177-1190, 2020.

[10] E. Kouloumpis, T. Wilson, and J. Moore, ‘Twitter Sentiment Analysis: The Good the Bad and the OMG!’, Proceedings of the International AAAI Conference on Web and Social Media , vol. 5, pp. 538-541, 2021.

[11] D. Zimbra, A. Abbasi, and D. Zeng, ‘The State-of-the-Art in Twitter Sentiment Analysis: A Review and Benchmark Evaluation’, ACM Transactions on Management Information Systems, vol. 9, pp. 1-29, 2018.

[12] T. Davidson, D. Warmsley, M. W. Macy, and I. Weber, ‘Automated Hate Speech Detection and the Problem of Offensive Language’, Proceedings of the International AAAI Conference on Web and Social Media, vol. 11, pp. 512-515, 2017.

[13] T. Davidson, D. Bhattacharya, and I. Weber, ‘Racial Bias in Hate Speech and Abusive Language Detection Datasets’, Proceedings of the Third Workshop on Abusive Language Online, pp. 25-35, 2019.

[14] W. Y. Ayele, ‘Adapting CRISP-DM for Idea Mining: A Data Mining Process for Generating Ideas Using a Textual Dataset’, International Journal of Advanced Computer Science and Applications, vol. 11, pp. 20-32, 2020.

[15] A. Pak and P. Paroubek, ‘Twitter as a Corpus for Sentiment Analysis and Opinion Mining’, Proceedings of the Seventh International Conference on Language Resources and Evaluation, vol.10, pp. 1320-1326, 2010.

[16] J. Ramos, ‘Using TF-IDF to Determine Word Relevance in Document Queries’, Proceedings of the First International Conference on Machine Learning, vol. 242, pp. 29-48, 2003.

[17] A. M. Kibrya, E. Frank, B. Pfahringer, and G. Holmes, ‘Multinomial Naive Bayes for Text Categorization Revisited’, AI 2004: Advances in Artificial Intelligence, pp. 488-499, 2005.

[18] M. Abbas, K. Ali, S. Memon, A. Jamali, S. Memon, and A. Ahmed, ‘Multinomial Naive Bayes Classification Model for Sentiment Analysis’, International Journal of Computer Science and Network Security, vol. 19, pp. 62-67, 2019.

[19] N. Jalal, A. Mehmood, G. S. Choi, and I. Ashraf, ‘A Novel Improved Random Forest for Text Classification Using Feature Ranking and Optimal Number of Trees’, Journal of King Saud University - Computer and Information Sciences, vol. 34, pp. 2733-2742, 2022.

[20] A. Poornima and K. S. Priya, ‘A Comparative Sentiment Analysis of Sentence Embedding Using Machine Learning Techniques’, Sixth International Conference on Advanced Computing and Communication Systems, pp. 493-496, 2020.

[21] J. Peng, K. Lee, and G. Ingersoll, ‘An Introduction to Logistic Regression Analysis and Reporting’, Journal of Educational Research, vol. 96, pp. 3-14, 2002.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, ‘Attention Is All You Need’, Advances in Neural Information Processing Systems, vol. 30, pp. 6000-6010, 2017.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 4171-4186, 2019.

[24] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, ‘RoBERTa: A Robustly Optimized BERT Pretraining Approach’, Computing Research Repository in arXiv, vol. 11692, 2019.