Tweet Hate Speech Classification and Analysis Using PySpark

Matthew Coleman, Nathan Mai, Brandon Shin, and Junyue Wang

10 June 2020

1 Abstract

While the highly developed internet provides convenience and privacy for users to speak freely, this environment can also be home to many malicious comments and racist messages. This brings to the forefront the complexity between balancing freedom of expression and defending human dignity. In this report, we aim to recognize the most commonly used hate speech words and bigrams from tweets, and predict whether a tweet is hate speech through machine-learning. We first briefly examine the dataset by summary graphs. Before moving to text analysis, we cleaned the tweets to clear unnecessary information in the raw dataset. We are working on a classification problem with labeled data, so we decide to apply Logistic Regression, Naive Bayes and Random Forest models to determine whether tweets are hate speech, offensive, or neither. To evaluate our supervised-learning models, we computed the model accuracy, and determined the best model to apply to our data was the Logistic Regression.

2 Introduction

Twitter is a social media platform where users can post and interact with messages known as “tweets”. This free-to-use platform allows anyone from around the world to tweet whatever they wish. This open nature means users come from different cultures, educational backgrounds, and political views, leading to differences in opinions. The internet anonymity and protection given by hiding behind a computer screen results in some users using their freedom to attack and harass others with derogatory and racial slurs. We attempt to attack this problem by using machine learning classification models to detect tweets as hate speech, offensive speech, or neutral tweets.

3 Data and Methods

3.1 Data

The dataset we used was the Hate Speech Tweet dataset from Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. The data comes as a CSV and as a pickled dataframe. No missing values or duplicated records were found. There are 24783 total observations and 6 main features in the dataset.

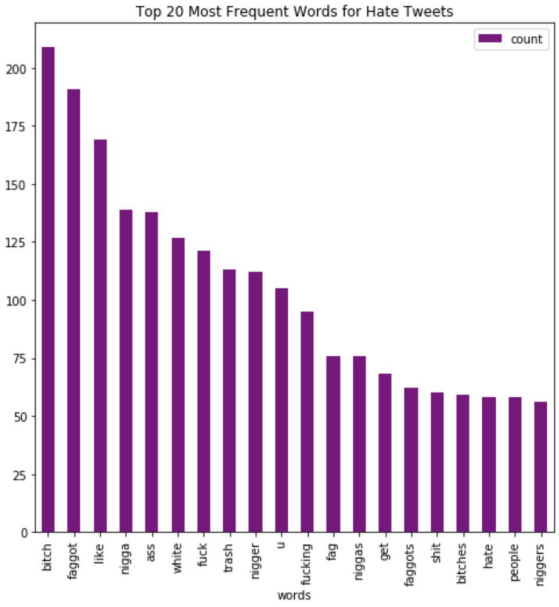
- count: The number of CrowdFlower (CF) users who coded each tweet, min value is 3.
- hate_speech: The number of CF users who labeled the tweet as hate speech.

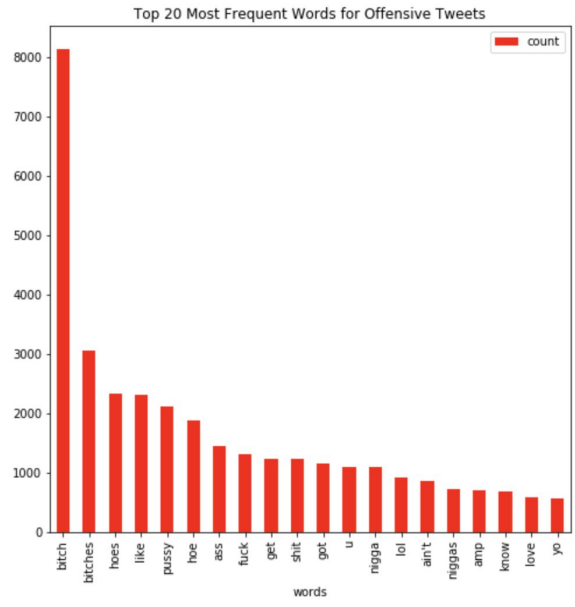
- offensive_language: The number of CF users who judged the tweet to be offensive.
- neither: The number of CF Users who judged the tweet to be neither offensive or non-offensive.
- class/label: Response, class label for the majority of CF users. 0 represents hate speech, 1 represents offensive language, and 2 represents neither. Directly below, we can see the counts by label.

label	count
0	1430
1	19190
2	4163

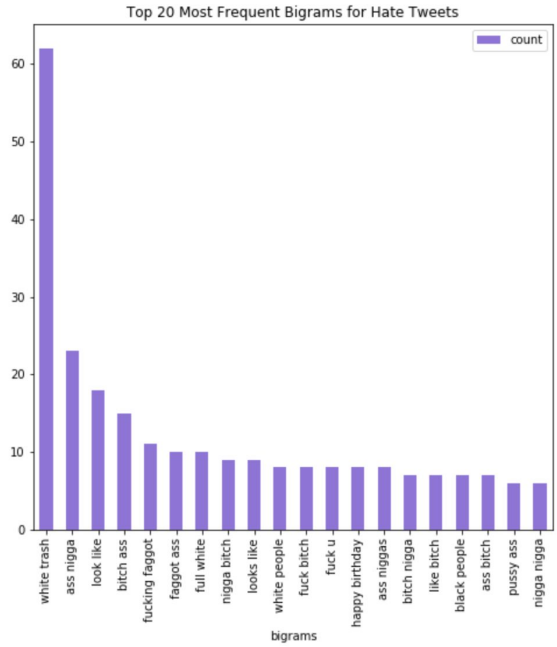
- tweets: The actual tweet posted by a Twitter user that was then classified by CF users as either hate speech, offensive language, or neither. This feature was then used to create tokens and bigrams, which were then used to create the tokens and bigrams used in our models.

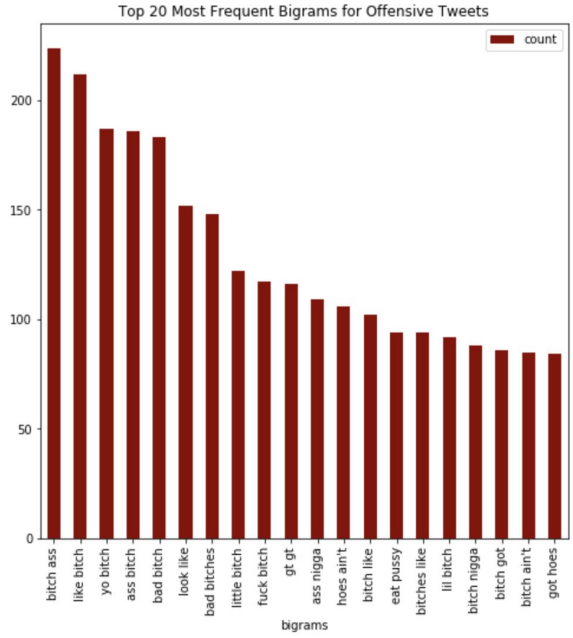
3.1.1 Most Common Word Counts for Hate and Offensive Tweets





3.1.2 Most Common Word Counts for Hate and Offensive Tweets





3.2 Methods

3.2.1 Data Cleaning and Feature Extraction

Using raw tweets which users from anywhere in the world could have posted introduced a number of challenges which required a high-degree of data cleaning before being used effectively in our Machine Learning Model.

Regex Replace
Using regular expressions, we were able to remove links, users' twitter handles, extra white spaces, special characters, and common phrases used in tweets such as "RT" for "retweet". This allowed us to look at the most important words.

Tokenizer
Using the tokenizer, we separated all the words into tokens, which we would be able to use in our machine learning models.
StopWordsRemover

The stop words remover allowed us to remove all the common stopwords, determined by the english stopwords in the pyspark.ml.feature library. Removing the stop words can remove commonly occurring words from the tweets, allowing for the tweets to have more distinct features, and therefore, higher accuracy.

NGram
Using NGram we were able to create bigrams from our tokens in order to see if particular combinations of words would improve the accuracy of our machine learning models.

TF-IDF
Using Term Frequency (TF) in our model gave us a metric of the frequency of words in the tweet. The Inverse Document Frequency (IDF) showed the frequency of the word across the entire document set. Using TF-IDF (multiplying these terms together) results in the TF-IDF score of the words. Higher scores represent a word more relevant to the tweet, and smaller scores represent a word less relevant. When using the TF-IDF transformation, we will be restricting our data to 10,000 features to improve our model run times.

3.2.2 Machine Learning Methods

Logistic Regression
Logistic regression is a linear classification method which computes the soft probability of a sample belonging to a certain class and then finds the optimal decision boundary to separate the classes. We will fit a logistic model on the tweet tokens and n-grams, using a multinomial classification output to predict whether a tweet is hate speech, offensive, or neither. We will use k-fold cross validation to tune the regression parameter and maximum iterations to maximize the classification accuracy.

Naive Bayes Classifier
Naive Bayes is a classification method based on Bayes' theorem that $p(C_k|\, x) = \frac{p(C_k)p(x|C_k)}{p(x)}$, where C_k corresponds to the K possible outcomes and $x = (x_1, \ldots, x_n)$ corresponds to the feature vector. This derives the probability of the given feature vector being associated with a label. Naive Bayes classifiers have a "naive" assumption of conditional independence for every feature. We will fit a Naive Bayes multinomial classifier on the tokens with stopwords removed and use k-fold cross validation to determine the best smoothing parameter. The best smoothing parameter will be determined by the cross-validated classification accuracy.

3.2.3 Model Validation and Cross Validation

We will train-test split our data into 75% (18587 observations) training and 25% (6193 Observations) testing. We did not use a validation set, but we will be evaluating the final model performance using the test set. We will use cross validation to retrieve the best parameters and use these parameters to train and test the final models. We tried Random Forest methods, but given the extremely long run times for high dimensional data and low model accuracies, we decided to exclude the model from our analyses. Tuning parameters through cross-validation resulted in poor model accuracies, further solidifying our decision. Because random forest splits on one feature, and we were working with categorical data (words), the random forest model was suboptimal.

4 Analysis and Discussion

4.1 Overview

In this part, we are going to investigate the hate speech dataset with different machine-learning methods. We analyze single words, with and without stopwords, and bigrams. Our results show that changing the analysis subject can change the results, so we had to carefully choose the preprocessed data for specific analysis. For example, bigrams provided analysis of two word combinations but resulted in the lowest accuracy. Our analysis suggested word combinations were not useful in determining whether a tweet contained hate speech when compared to single tokens. Also, all of our models had difficulty differentiating offensive language and hate speech, further reducing the accuracy. After analyzing all models, we will select the most appropriate model.

4.2 Logistic Regression (Stopwords Removed)

Training Set

Confusion Matrix

label_prediction	0.0	1.0	2.0
0	376	562	122
1	219	13829	383
2	18	215	2901

Accuracy by Label

Hate	Offensive	Neither
0.613377	0.946803	0.851732

Overall Accuracy: 0.912884

Test Set

Confusion Matrix

label_prediction	0.0	1.0	2.0
0	104	227	39
1	109	4477	173
2	17	119	893

Accuracy by Label

Hate	Offensive	Neither
0.452174	0.92826	0.808145

Overall Accuracy: 0.882845

The logistic regression model with stop words removed had an overall training accuracy of 91.29% and a test accuracy of 88.28%. The training and test accuracies are similar, which tells us the original model may be fit well to the data. Observing the accuracy by label tells us that the accuracy for both the training and test set is highest for offensive tweets and neither tweets. While these two are highly accurate, the training accuracy for hate tweets is slightly better than a coin-flip, while the test accuracy is slightly worse than a coin-flip. One possible reason for this is the class imbalance of the dataset. Having a very small fraction of the data being hate tweets could have resulted in there being a small amount of hate tweets for the model to be fit on and a large amount of other tweets to be fit.

4.3 Logistic Regression (Bigrams)

Training Set

Confusion Matrix

label_prediction	0.0	1.0	2.0
0	1005	54	1
1	3	14419	9
2	3	49	3082

Accuracy by Label

Hate	Offensive	Neither
0.994065	0.992907	0.996766

Overall Accuracy: 0.993570

Test Set

Confusion Matrix

label_prediction	0.0	1.0	2.0
0	42	262	66
1	253	3691	815
2	62	615	352

Accuracy by Label

Hate	Offensive	Neither
0.117647	0.808012	0.285483

Overall Accuracy: 0.670606

The logistic regression model on bigram data was extremely overfit to the data, with near 100% accuracy on all labels for the data, and an overall accuracy of 99.35%. The test accuracies of the hate and neither tweets were very poor, with the hate and neither accuracy being 11.76% and 28.54%, respectively, and the overall accuracy being 67.06%. It is possible, because the bigrams can be very unique, the original data may have been ankle to be easily predicted with the model, but once the model was introduced to new bigrams which were not in the vocabulary, it had issues being able to classify the tweets accurately. The offensive tweets were the most accurate, with a 80.8% accuracy on the test set. As with the normal LR model, there were many more tweets, allowing for the model to see more data points and create a less biased model.

4.4 Logistic Regression (Stopwords Included)

Training Set

Confusion Matrix

label_prediction	0.0	1.0	2.0
0	1047	29	0
1	32	14371	4
2	0	5	3137

Accuracy by Label

Hate	Offensive	Neither
0.970343	0.99764	0.998727

Overall Accuracy: 0.996244

Test Set

Confusion Matrix

label_prediction	0.0	1.0	2.0
0	90	222	42
1	373	4087	323
2	53	293	675

Accuracy by Label

Hate	Offensive	Neither
0.174419	0.888092	0.649038

Overall Accuracy: 0.796986

The logistic regression model for the tweets with the stopwords included ended up being very similar to the LR bigram model, with very high training accuracies, and an overall training accuracy of 99.62%. The test accuracy was 79.7%, suggesting this model was overfit to the data. Leaving stop words in the tweets can result in common words appearing in all the tweets, making it difficult to discern tweets from each other.

4.5 Naive Bayes (Stopwords Removed)

Training Set

Confusion Matrix

label_prediction	0.0	1.0	2.0
0	467	490	103
1	472	13441	518
2	58	763	2313

Accuracy by Label

Hate	Offensive	Neither
0.468405	0.914727	0.788344

Overall Accuracy: 0.869271

Test Set

Confusion Matrix

label_prediction	0.0	1.0	2.0
0	124	216	30
1	150	4432	177
2	20	244	765

Accuracy by Label

Hate	Offensive	Neither
0.421769	0.905969	0.787037

Overall Accuracy: 0.860004

The Naive Bayes classifier was the second best model with an overall training accuracy of 86.93%, and an overall test accuracy of 86.04%. The accuracy by label was similar for both the training and test set, with hate tweets having 46.84% and 42.18% for the training and test sets. The offensive accuracy was approximately 90% for both sets, and the neither tweets had an accuracy of approximately 78%. The training and test error are very similar, which suggests the model may be well fit to the data.

5 Conclusion

5.1 Model Training and Test Accuracies

Models	Training Accuracy	Test Accuracy
Logistic Regression (No Stopwords)	0.912884	0.882845
Logistic Regression (Bigrams)	0.993570	0.670606
Logistic Regression (Stopwords Included)	0.996244	0.796986
Naive Bayes (No Stopwords)	0.869271	0.860004

5.2 Final Model

Final Model

Confusion Matrix

label_prediction	0.0	1.0	2.0
0	104	227	39
1	109	4477	173
2	17	119	893

Accuracy by Label

Hate	Offensive	Neither
0.452174	0.92826	0.808145

Overall Accuracy: 0.882845

The final model used will be the Logistic Regression model with no stopwords. This model had the highest test accuracy of 88.28%, and did not have a high training accuracy, suggesting an overfit model. While the overall accuracy of the model is good, especially for the offensive or neither tweets, the hate classification is not very useful. At 45.22%, the classifier is worse than a coin flip to determine whether a tweet contains hate speech. The low accuracy of the hate tweet classification could be a result of the number of observations in the “hate” class. The offensive tweet class, which had the highest accuracy for all of the models had 19,000 observations, where the hate tweets only had 1,400 observations. The neither class had 4,100 observations, and was almost twice as accurate as the hate tweet classification. If this assumption were true, a small

increase in the number of data points in the “hate” class could result in a much better final model. This class imbalance also makes the overall accuracy rate skewed to be higher because $Accuracy = \frac{Number\ of\ True\ Positives}{Number\ of\ Observations}$, so if there are a large number of true positives in the “offensive” class, then the overall accuracy of the algorithm is going to be similar to the accuracy of the class with the largest number of observations, and as the number of observations increases, it will become more skewed. This can be improved by collecting more hate tweets and training the model on more observations.

5.3 Future Research

Although we applied some machine learning processes to categorize hate speech, there are more areas of exploration in the field of social media speech-type analysis. In our analyses, we removed the twitter function notes for replies or retweets and analyzed the pure contents of the tweet. However, by examining the original post, we can understand which topics are most likely to result in hate speech. Analyzing the context of a post could better screen posts with sensitive phrases or words and prevent hate speech. Further, we did not take into consideration newly emerged “online words,” which can contain combined letters, emojis or numbers. Users could use “online words” such as b1tch, fxxk, or f4ggot to attack users and avoid system detection of hate words. We believe exploration into these newly formed “online words” would be beneficial to accurate hate speech recognition. If specific users were frequent abusers of hate speech, an auto warning function which can flag the users content as offensive or even block the users could be one field of exploration. This can be analogized to what is being done on twitter with the president's controversial content, except automated through the use of machine learning. We could go further in depth with our analyses and label tweets in the neither category as “neutral” and “positive.” While this would make our model more complex, it would allow for classification of more precise categories, and result in a more informative model. Our research could also be “flipped” and be made to classify positive speech to positive speech (instead of label 0 = hate speech, 0 = positive speech).

6 References

[1] https://github.com/t-davidson/hate-speech-and-offensive-language

[2] https://monkeylearn.com/blog/what-is-tf-idf/

[3]https://dataespresso.com/en/2017/10/24/comparison-between-naive-bayes-and-logistic-regression/

[4] https://en.wikipedia.org/wiki/Naive_Bayes_classifier