UCI Paul Merage School of Business
Leadership for a Digitally Driven World™

BANA 273: Machine Learning Analytics Project Report

Adwait Toro (89487135)
Vedant Kshirsagar (50236220)
Sushrut Shendre (76883110)
Sneha Lakshmikanthaiah (37889217)
Rishab Ballekere Narayana Gowda (57844726)
Tanvir Brar (36241934)
Chavi Singal()

Abstract

Toxic online content has become a major issue in today's world due to an exponential increase in the use of internet by people of different cultures and educational background. Differentiating hate speech and offensive language is a key challenge in automatic detection of toxic text content. In this report, we propose an approach to automatically classify tweets on Twitter into two classes: hate speech and non-hate speech. Using Twitter dataset, we perform experiments by leveraging bag of words and the term frequency-inverse document frequency (TFIDF) values to multiple machine learning models. We perform comparative analysis of the models considering both of these approaches. After tuning the model giving the best results, we achieved 89% accuracy and 84% recall from the Logistic Regression model upon evaluating it on test data. We also create a module using flask which serves as an MVP of our approach.

Overview

The goal of our project was to classify tweets into two categories, hate speech or non-hate speech. Our project analyzed a dataset CSV file from Kaggle containing 31,935 tweets. The dataset was heavily skewed with 93% of tweets or 29,695 tweets containing non-hate labeled Twitter data and 7% or 2,240 tweets containing hate-labeled Twitter data. The first step of the project was to downscale to remove the bias the dataset inherently contains. Our approach utilized 2 main models, Bag of words and Term Frequency Inverse Document Frequency (TFIDF). The bag-of-words model is a simplified representation used in natural language processing and information retrieval. In this model, a text, such as a sentence or a document, is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. TFIDF, on the other hand, is a numerical statistic that is intended to reflect how important a word is to a document in a collection. Before we dove into these models, we had to clean our data as tweets contain many different tenses, grammatical errors, unknown symbols, hashtags, and Greek characters. To do this we employed lemmatization, stemming, stop words, and omission. Lemmatization removes the inflectional endings of words and returns the word to the base or dictionary form of it itself. Stemming is similar to lemmatization in that it reduces the inflected or derived words to their word stem. A stop word is a commonly used word, such as "the", "a", "an", "in", that we programmed to ignore as it holds no importance. Lastly, we omitted any foreign characters and Greek symbols so our dataset could be ready to analyze.

Inspiration

The inspiration to explore this topic came from both the increase in the instances of hate speech and hate speech awareness. According to the New York Times, "Personal attacks motivated by bias or prejudice reached a 16-year high in 2018, with a significant upswing in violence against Latinos outpacing a drop-in assault targeting Muslims and Arab-Americans." This rise can also be attributed to the rise of digital social media platforms which make hate speech much simpler to spread. As a whole, 59.6 % of all hate crime incidents involve race and ethnicity bias. The next highest bias was religion which accounted for 18.7% of all incidents. Our research validated these results as we saw most of our top 20 negative words mention race or religion. These instances of vitriol can have immense negative impacts on our society and especially our younger generation. According to public research, 22% of respondents say they feel less safe in their community because of online hate speech and over 85% support setting up a task force to monitor cyber bullying. The issue has become so poignant now that the first lady of the United States, Melania Trump, has made it her main goal to combat cyberbullying and hate speech online. As the statistics show, this issue is very important to our community and further research must be done in recognizing hate speech and doing more to combat it.

Data

We collected the data from Kaggle.
(Source: https://www.kaggle.com/vkrahul/twitter-hate-speech)

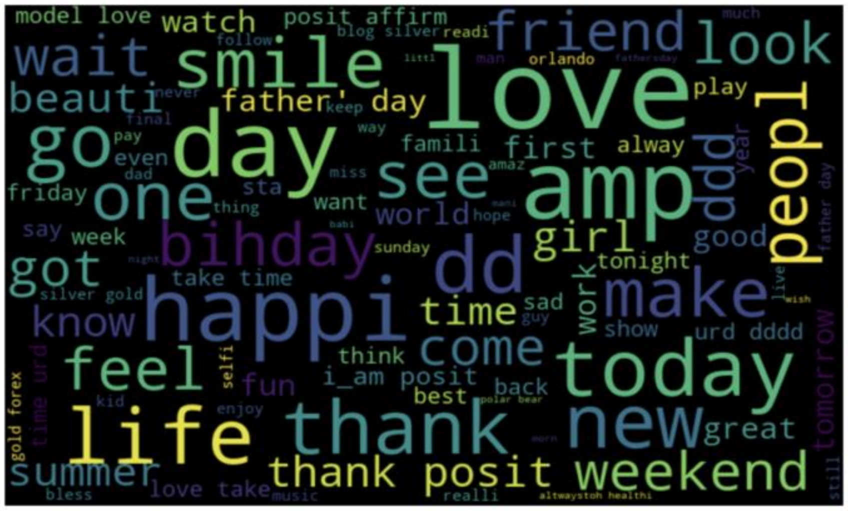
Data Dictionary

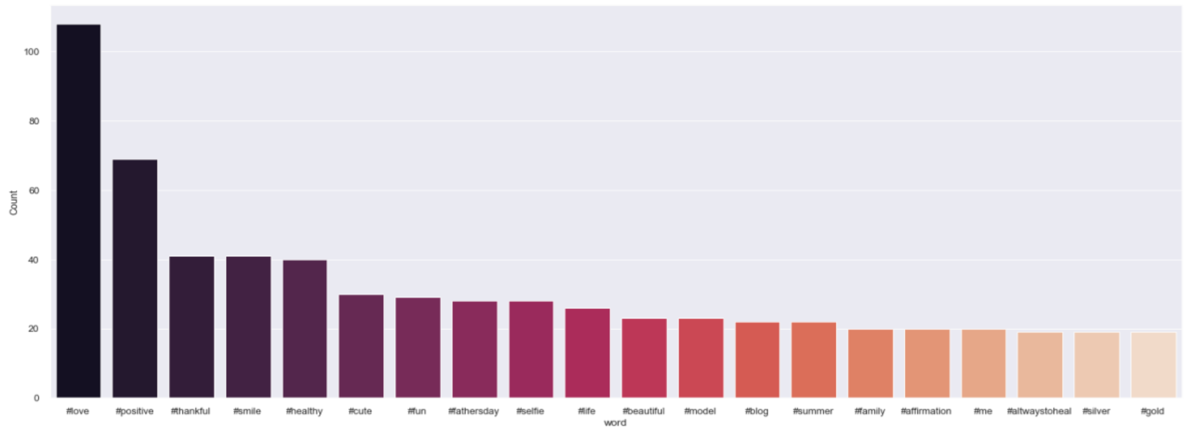
Field Name	Description
id	Serial Number
label	Class of the tweet: 1-hate tweet; 0-good tweet
tweet	The Tweet content

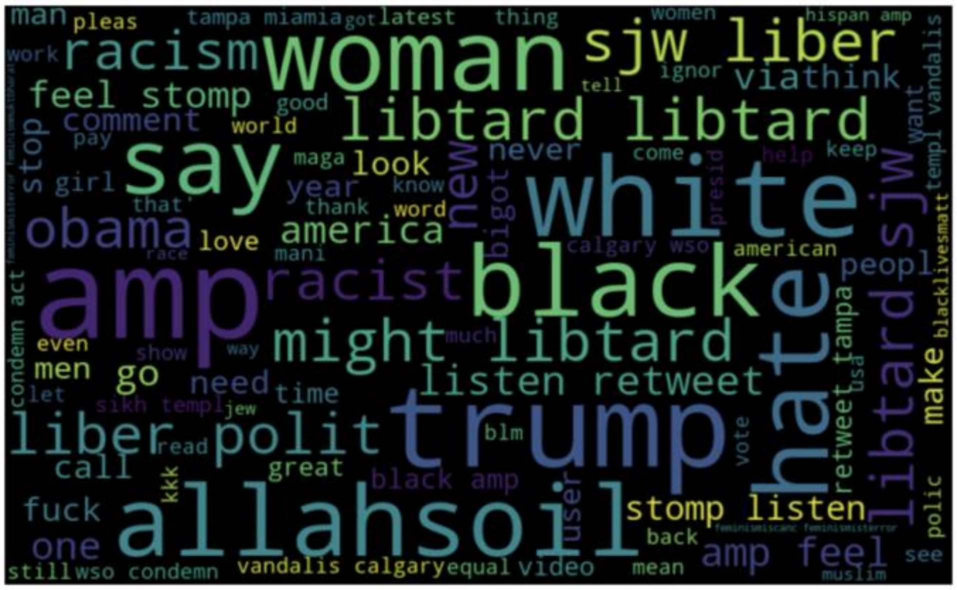
Visualizations and Word Clouds

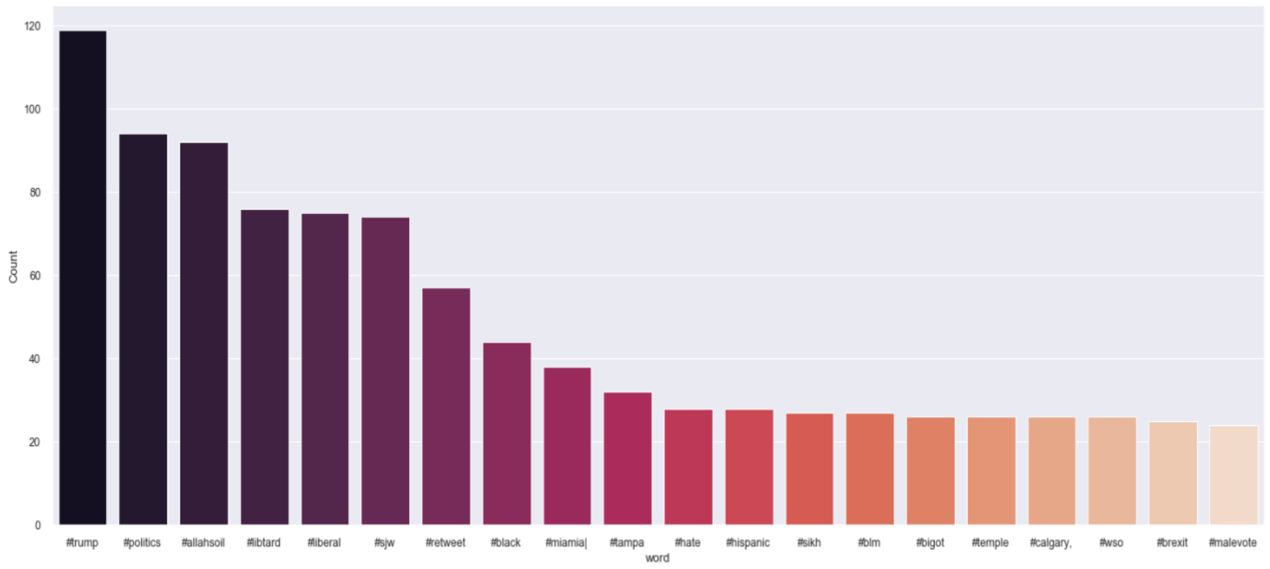
We created a word cloud to get a sense of the words being repeated the most in all the tweets. We did this for both the positive and the negative sentiment carrying words. Post this we created a bar graph to analyze the difference in the usage frequency between the top used words in both the positive as well as the negative sentiment. Below are the results.

Positive









Data Architecture

As discussed earlier, the data we have is imbalanced. We have a total of 31935 rows out of which the hate tweets comprise of only 7% whereas the good tweets comprise of 93% of the total tweets. Modelling on such an imbalanced data would not be the best way to go forward since the model won't be able to learn what pieces of text or information causes the tweet to be classified as a hate tweet.

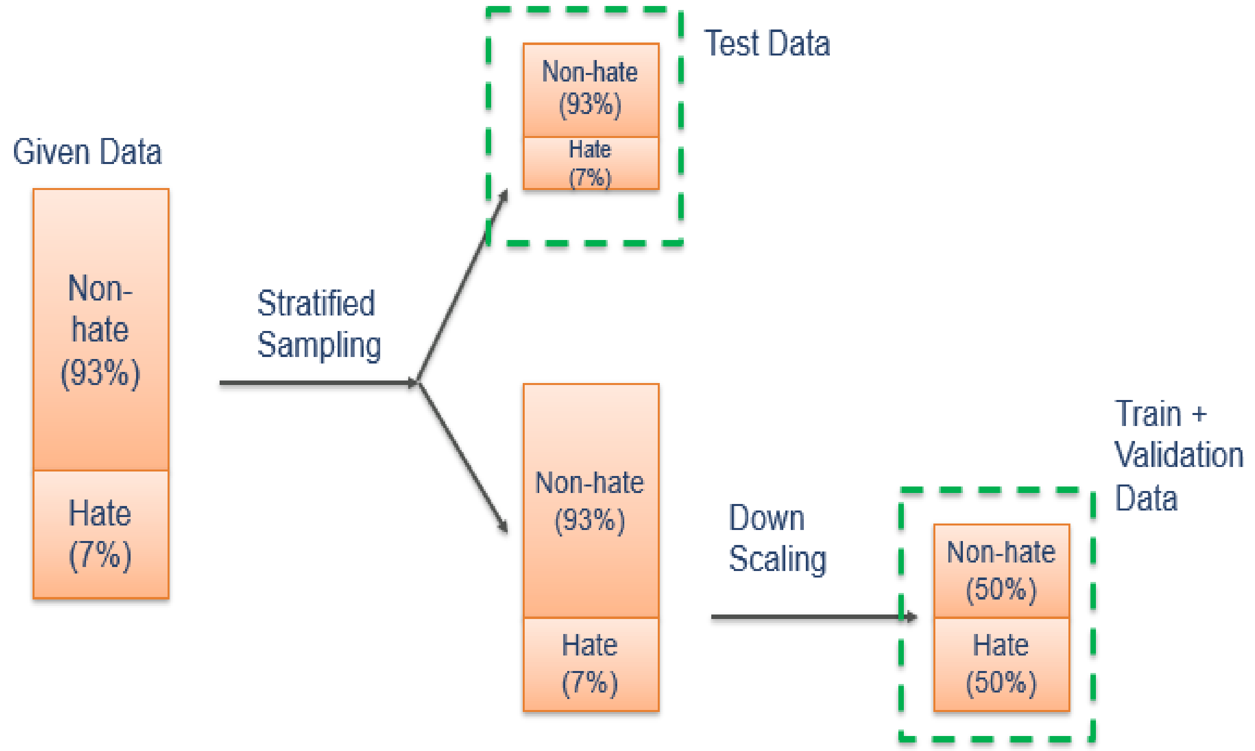
Even if we go ahead with modelling on such data (which we did try), the results will be misleading. In such imbalanced datasets, due to lack of learning, the model would simply predict each tweet as a good tweet. When it does so, although the accuracy metric is very high, but it is not doing a good job of classification since it is not able to classify the hate tweets correctly.

Therefore, we perform strategic sampling and separate the data into a test set and another temporary set. Note that since we have done strategic sampling, the ratio of good tweets to hate tweets is 93:7 for both the test and temporary datasets.

On the temporary data, we first tried to perform up sampling using SMOTE (Synthetic Minority Oversampling Technique). Since the SMOTE packages don't work directly for textual data, we wrote our own code for it. The process was as follows:
We took a subset of all the hate tweets in the temporary data (s1). We formed a corpus of all the unique words within them. We also made a list containing the number of words in each tweet.
We then made a dataset having the number of rows equal to that in the subset s1. Now, for each row, we chose a random number from the list (say n1) which would essentially be the number of words in that row. Then we chose n1 words at random from the corpus and placed them in that row. We repeat the same procedure for other rows.

We then repeated these multiple times until the number of rows in this synthetic data was at par with the number of good tweets we had in our temporary data. But when we employed the Bag of Words approach for feature generation (we will discuss this in detail later), the number of features went up to around 100,000. Since our machines were not capable of doing that, we discarded the SMOTE oversampling method.

We instead decided to down sample. We took a subset of only the non-hate tweets from the temporary dataset. From this, we selected *n* random samples, where *n* is the number of hate tweets in the temporary data. We then joined this with the subset of hate tweets in the temporary data. This dataset is now the training data that we use for our feature generation and modelling purposes. Remember that the test data is still in a 93:7 ratio of good tweets to hate tweets, since this is what the real-time data would be like.



Approaches

We have looked at two major approaches for feature generation: Bag of Words (BOW) and Term Frequency Inverse Document Frequency (TFIDF).

1. Bag of Words

2. Term Frequency Inverse Document Frequency

Bag of words

A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms.

It is called a "*bag*" of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document. The intuition is that documents are similar if they have similar content. Further, that from the content alone we can learn something about the meaning of the document.

The objective is to turn each document of free text into a vector that we can use as input or output for a machine learning model. Because we know the vocabulary has 10 words, we can use a fixed-length document representation of 10, with one position in the vector to score each word. The simplest scoring method is to mark the presence of words as a Boolean value, 0 for absent, 1 for present.

Using the arbitrary ordering of words listed above in our vocabulary, we can step through the first document ("*It was the best of times* ") and convert it into a binary vector.

The scoring of the document would look as follows:

- "it" = 1
- "was" = 1
- "the" = 1
- "best" = 1
- "of" = 1
- "times" = 1
- "worst" = 0
- "age" = 0
- "wisdom" = 0
- "foolishness" = 0

As a binary vector, this would look as follows:

```
1 [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
```

The other three documents would look as follows:

1 "it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]

2 "it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]

3 "it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

If your dataset is small and context is domain specific, BoW may work better than Word Embedding. Context is very domain specific which means that you cannot find corresponding Vector from pre-trained word embedding models (GloVe, fastText etc.)

TFIDF

TF*IDF is an information retrieval technique that weighs a term's frequency (TF) and its inverse document frequency (IDF). Each word or term has its respective TF and IDF score. The product of the TF and IDF scores of a term is called the TF*IDF weight of that term.

Put simply, the higher the TF*IDF score (weight), the rarer the term and vice versa.

The TF*IDF algorithm is used to weigh a keyword in any content and assign the importance to that keyword based on the number of times it appears in the document. More importantly, it checks how relevant the keyword is throughout the web, which is referred to as *corpus*.

For a term t in a document d, the weight Wt, d of term t in document d is given by:

$$W_{t,d} = TF_{t,d} \log (N/DF_t)$$

Where:

- $TF_{t,d}$ is the number of occurrences of t in document d.
- DF_t is the number of documents containing the term t.
- N is the total number of documents in the corpus.

How is TF*IDF calculated? The TF (term frequency) of a word is the frequency of a word (i.e. number of times it appears) in a document. When you know it, you're able to see if you're using a term too much or too little.

For example, when a 100-word document contains the term "cat" 12 times, the TF for the word 'cat' is

$$TF_{cat} = 12/100 \text{ i.e. } 0.12$$

The IDF (inverse document frequency) of a word is the measure of how significant that term is in the whole corpus.

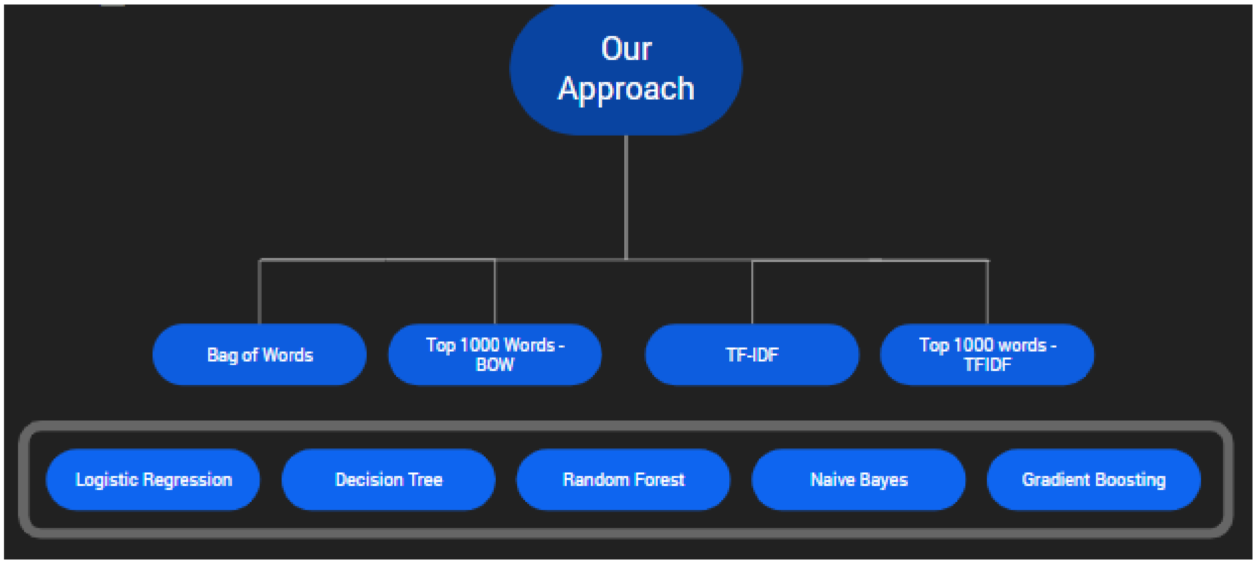
Data Cleaning

Languages we speak and write are made up of several words often derived from one another and can contain a lot of words which don't add a lot of meaning or context. In order to normalize the data, we implemented 5 approaches for data cleaning. They are as follows,

1. **Stop words removal**: Stop words are usually articles or prepositions which do not help us to find the context or the true meaning of a sentence. These are words that can be removed without any negative consequences to the final model that you are training. Commonly used in English language include "is"," and"," are" etc.
2. **Greek characters**: We next removed special characters and numbers. Numbers rarely contain useful meaning. Special characters can bloat our term-frequency matrix.
3. **Slang words**: Slang words include informal short forms of words which are usually used in speech. Due to lack of an existing dictionary, we mapped a few slang words to their original forms such as "luv" to love, "thx" to thanks to mention a few examples.
4. **Stemming**: With stemming, words are reduced to their word stems by cutting off the end or the beginning of the word, taking into account a list of common prefixes and suffixes that can be found in an inflected word. A word is looked at and run through a series of conditionals that determine how to cut it down. We used the porter stemmer algorithm to achieve the same.
5. **Lemmatization**: Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words. So, it links words with similar meaning to one word. For example, *runs, running, ran* are all forms of the word *run*, therefore *run* is the lemma of all these words.

Modelling

For both the Bag of words and TFIDF, we run 5 classification algorithms, namely Logistic Regression, Naive Bayes, Decision Tree, Random Forest and Gradient Boosting. Furthermore, we also perform dimensionality reduction in both the approaches and again run these 5 algorithms.



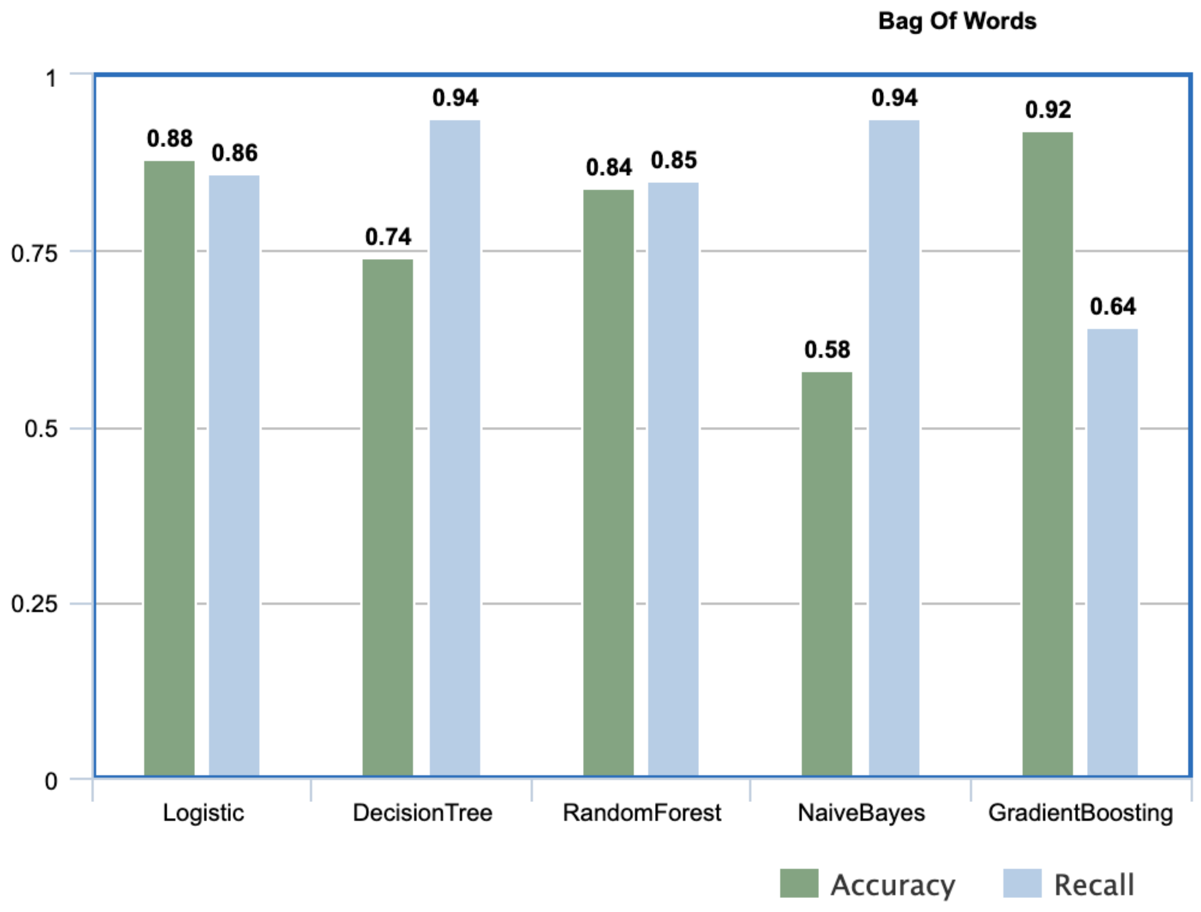
i) We implement the Bag of Words for feature generation, on our training dataset. In this approach, we try to find the probability that a tweet is a hate tweet or not, given the presence of certain words and their frequency in the tweet.

Our training data is a total of 4480 tweets which has a corpus of 12566 unique words. After implementing the Bag of Words feature engineering, we end with a dataframe having 4480 rows and 12566 columns, where each column corresponds to a unique word. We then concatenate the label as a column.

When we run the classification algorithms, we get the following results:

Algorithm	Accuracy	Recall
Logistic Regression	0.88	0.86
Decision Tree	0.74	0.94
Random Forest	0.84	0.85
Naive Bayes	0.58	0.94
Gradient Boosting	0.92	0.64

To better visualize these results, we plot the following cluster column chart.

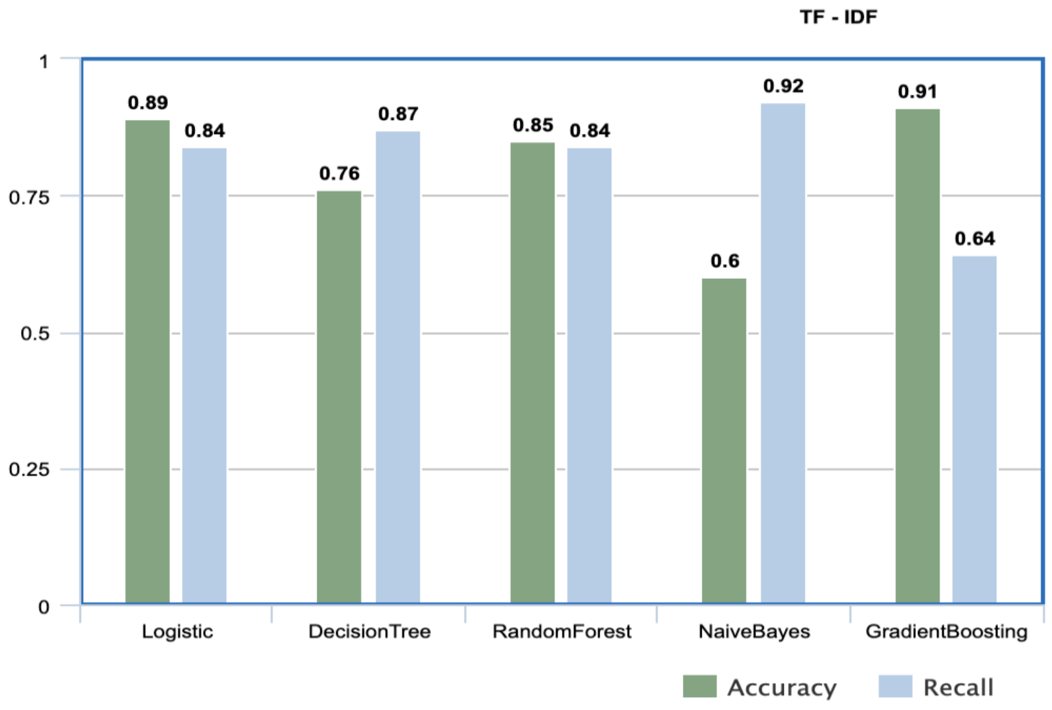


We now debate over the reasoning of having accuracy and recall as our metrics and how they'll help us choose our champion model. We first looked at the overall accuracy and according to this, gradient boosting is the best model. But as we discussed in the data architecture, our test data contains data in a 93:7 ratio for good tweets to hate tweets. Thus, having a high accuracy may not be the best measure. Thus, we looked at the recall, which essentially captures the proportion of hate tweets we are able to capture. If this number is high, it would mean that we are correctly flagging the actual hate tweets as hate tweets and thus achieving our objective. Looking at the chart above, Naive Bayes would be the best model to choose. But we forget one more aspect to this project. All the tweets we capture as hate tweets would be forwarded to an operations team which would review each of these tweets individually. Therefore, having a large number of false positives will mean that the analysts will have to unnecessarily scrutinize a lot of tweets which are actually good tweets. This will lead to inefficient utilization of time, resources and personnel. Thus, our recommendation is to have a good recall but at the same time ensure that the number of false positives are kept to a minimum. Thus, it would be ideal to move forward with models which are giving good scores for both accuracy and recall. Thus, we select logistic regression and random forest as the models which we may go forward with.

ii) Term Frequency Inverse Document Frequency (TFIDF)

There are a lot of words which are actually slang or abusive words but are so common that people use them a lot in their daily lives. So, if there is a tweet containing these words, it may not actually be intended as a hate tweet. To draw the line between whether a slang/abusive word is intended as hate or not, we tried the TFIDF approach since it allocates a low weightage to such words. So, instead of the Bag of Words model where we have a count of how many times that word occurred in the tweet, this will have a TFIDF weight instead. After running the algorithms, we obtain the following results.

Algorithm	**Accuracy**	**Recall**
Logistic Regression	0.89	0.84
Decision Tree	0.76	0.87
Random Forest	0.85	0.84
Naive Bayes	0.60	0.92
Gradient Boosting	0.91	0.64



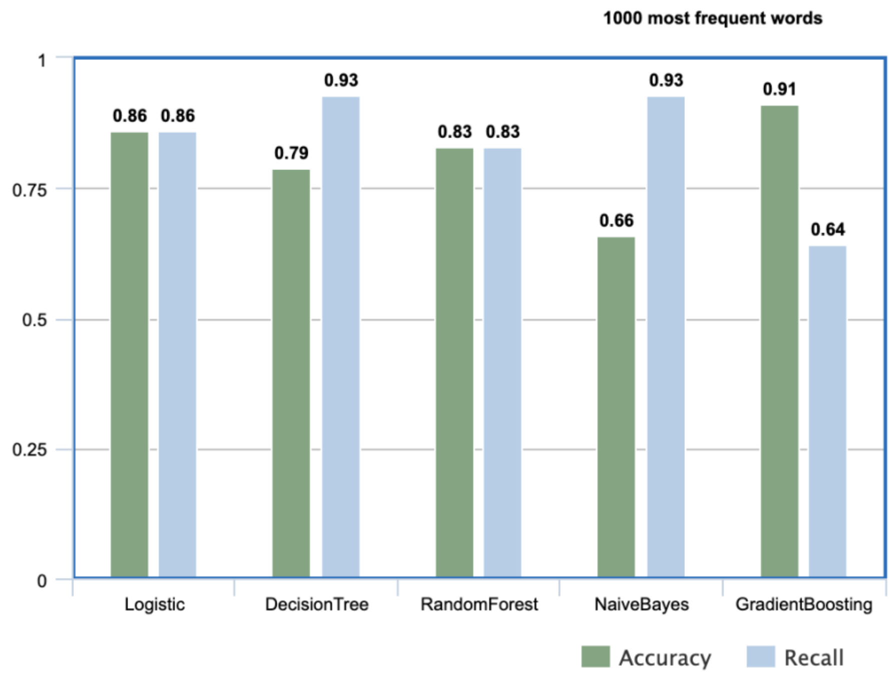
Again, logistic regression and random forests give us good results homogeneously for accuracy as well as recall.

iii) Bag of Words and TFIDF with 1000 most frequent words

Since the number of features was equal to the number of unique words and hence very large (12566), we hypothesized that the two models we have discussed so far could be overfitting. Therefore, we try to reduce the dimensionality by taking the 1000 most frequent words (and hence variables). After this variable selection, we implement the classification algorithms again. We do this for both Bag of Words as well as TFIDF. The results obtained are tabulated and plotted as follows:

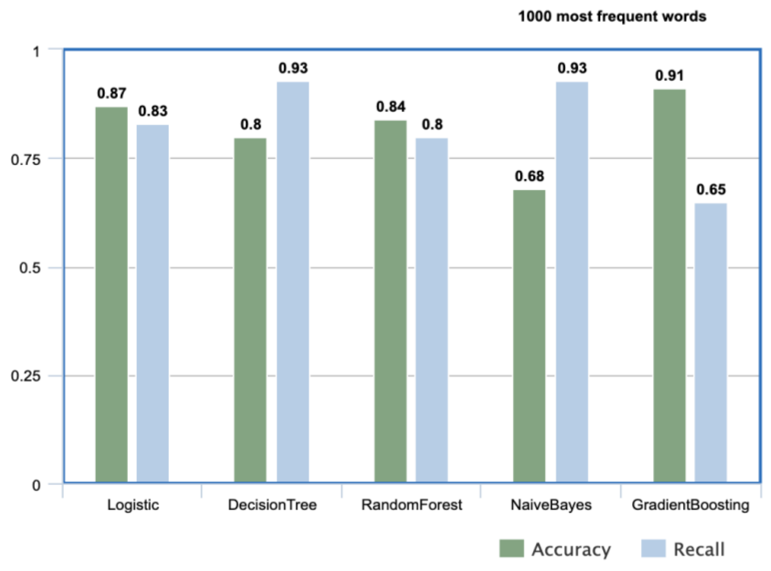
<u>Bag of Words with 1000 most frequent words only</u>

Algorithm	**Accuracy**	**Recall**
Logistic Regression	0.86	0.86
Decision Tree	0.79	0.93
Random Forest	0.83	0.83
Naive Bayes	0.66	0.93
Gradient Boosting	0.91	0.64



<u>TFIDF with 1000 most frequent words only</u>

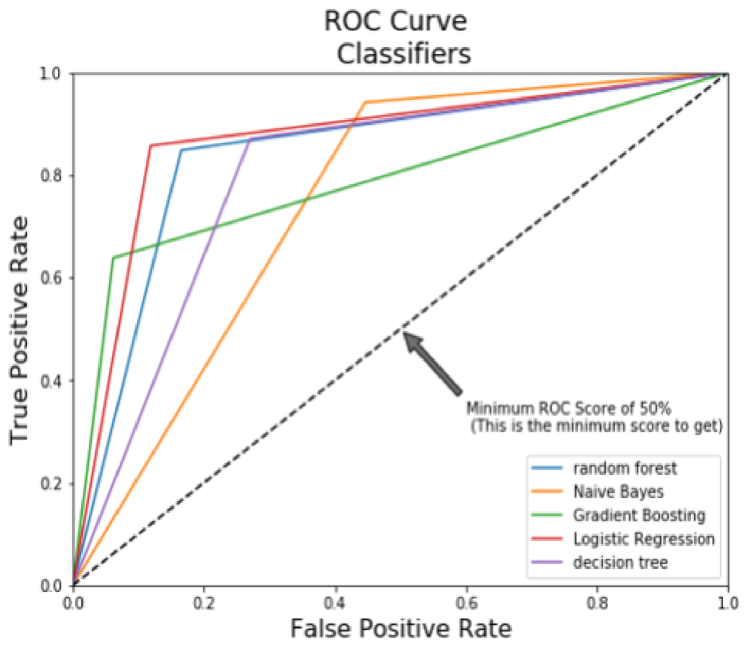
Algorithm	**Accuracy**	**Recall**
Logistic Regression	0.87	0.83
Decision Tree	0.80	0.93
Random Forest	0.84	0.80
Naive Bayes	0.68	0.93
Gradient Boosting	0.91	0.65



The results obtained after running the models with reduced dimensions were consistent with those without dimensionality reduction. Gradient Boosting again gives a good accuracy; naive bayes again gives a good recall; and logistic regression and random forest give good results for both of these metrics.

However, there are no significant improvements for these models. While the metrics improve for some algorithms, they depreciate for others and the amount by which they improve/reduce is very less.

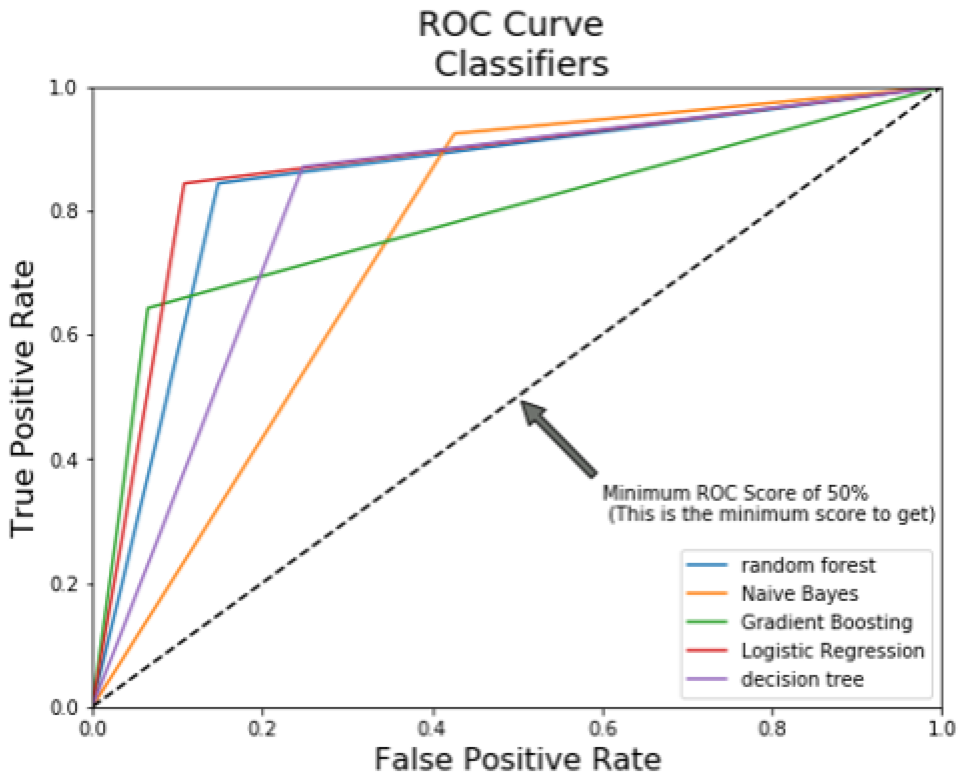
Lastly, we look at the ROC curves. Following is the curve for the Bag of Words approach (all words):



The following table shows the Area under the curve (AUC) for the various algorithms

Algorithm	**AUC**
Logistic Regression	0.869
Naive Bayes	0.748
Random forest	0.842
Decision Tree	0.800
Gradient Boosting	0.788

Now, let us look at the ROC curves and AUC table for the TFIDF approach (all words)



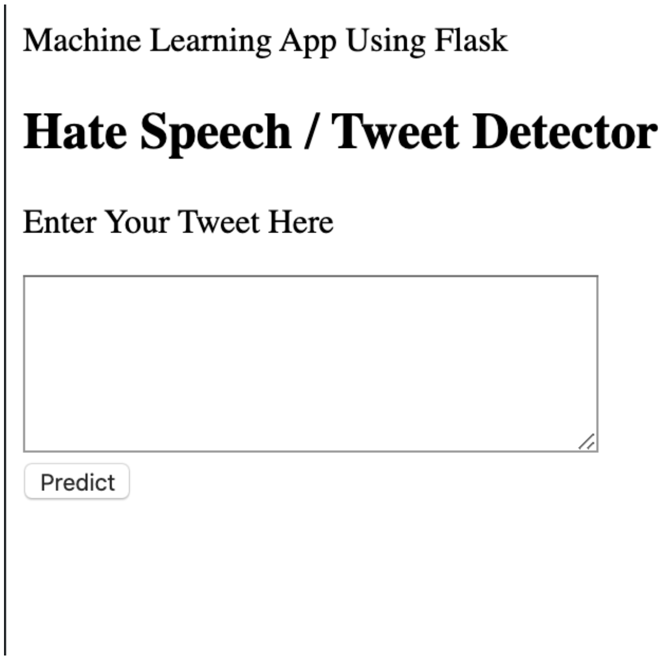
Algorithm	AUC
Logistic Regression	0.868
Naive Bayes	0.749
Random forest	0.848
Decision Tree	0.811
Gradient Boosting	0.789

As we can see, logistic regression gives us the best AUC. Also, if we look at its ROC curve, we can see that it covers the False Positive Rate (FPR) and True Positive Rate (TPR) area equally. Compare this to Naive Bayes which shows a poor false positive performance, which was also evident by its accuracy score earlier. Similarly, gradient boosting has a poor true positive performance which was evident by its low recall score.

In addition, logistic regression is very explainable in nature, i.e. the model can be explained with the help of a simple exponential equation, and its coefficients can be visualized and compared. Therefore, we recommend logistic regression as the best model for our case.

MVP

After all the data cleaning, feature engineering and modeling, we decided to deploy the model to yield something useful from this project. The main objective of this project was to develop a system that can detect the hate tweets so that Twitter could take appropriate actions and reduce the causes of social bullying. Thus, we deployed our machine learning model using Flask technology.
Below is a snapshot of the website: -


Machine Learning App Using Flask

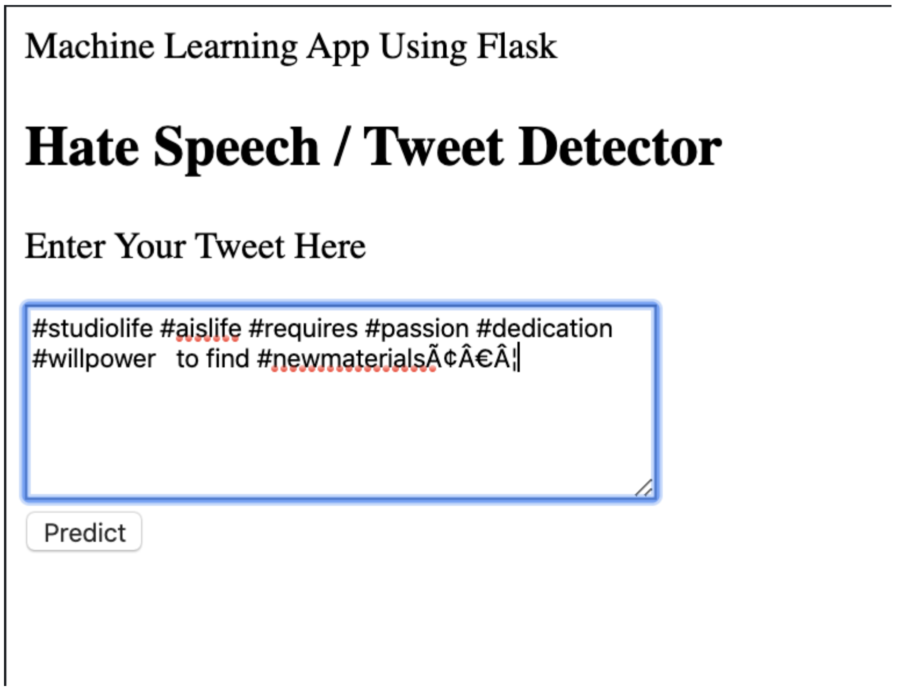
Hate Speech / Tweet Detector

Enter Your Tweet Here

The website was built using HTML and CSS wherein a person can enter a tweet given in the text box on the website.

Consider 2 use cases of our proposed model: -

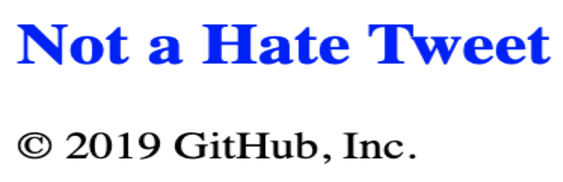
1 - The user enters a positive tweet and then correctly gets classified as a positive tweet.


Machine Learning App Using Flask

Hate Speech / Tweet Detector

Enter Your Tweet Here

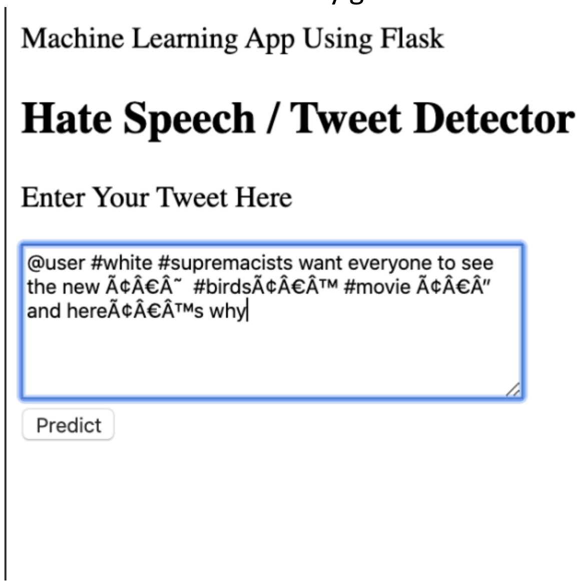
#studiolife #aislife #requires #passion #dedication #willpower   to find #newmaterialsÃ¢Â€Â¦



Not a Hate Tweet

© 2019 GitHub, Inc.

2 - The user enters a hate tweet and then correctly gets classified as a hate tweet.

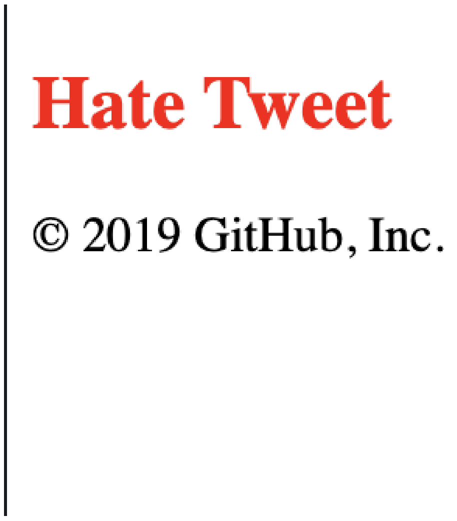


Machine Learning App Using Flask

Hate Speech / Tweet Detector

Enter Your Tweet Here

@user #white #supremacists want everyone to see the new Ã¢Â€Â˜ #birdsÃ¢Â€Â™ #movie Ã¢Â€Â" and hereÃ¢Â€Â™s why



Hate Tweet

© 2019 GitHub, Inc.

Key Insights and Learnings

Politics, race and sexual tweets form a major chunk of hate tweets.
Results obtained when we played around with an imbalanced data set were inaccurate. The model predicts new data to belong to the major class in the training set due to the skewed nature of the train data.
The weighted accuracy of a classifier is not the only metric to be looked at while evaluating the performance of the model. Business context plays a vital role as well.

Challenges and Future Steps

A challenge faced by automatic hate speech detection systems is the changing of attitudes towards topics over time and historical context. For example, in the following excerpt:

"…The merciless Indian Savages, whose known rule of warfare, is an undistinguished destruction of all ages, sexes and conditions…"

Intuition suggests that this is hate speech; it refers to Native Americans as "merciless Indian savages" and dehumanizes them by suggesting that they are inferior. Indeed, the text satisfies conditions used in most definitions of hate speech. However, this text is actually a quote from the Declaration of Independence. Given the historical context of the text, the user who posted it may not have intended the hate speech result, but instead meant to quote the historical document for other purposes. This shows that user intent and context play an important role in hate speech identification.

As another example, consider the phrase "the Nazi organization was great." This would be considered hate speech because it shows support for a hate group. However, "the Nazi's organization was great" isn't supporting their ideals but instead commenting on how well the group was organized. In some contexts, this might not be considered hate speech, e.g., if the author was comparing organizational effectiveness over time. The difference in these two phrases is subtle but could be enough to make the difference between hate speech or not.

Another remaining challenge is that automatic hate speech detection is a closed-loop system; individuals are aware that it is happening, and actively try to evade detection. Users who desire to spread the hateful messages quickly find ways to circumvent these measures by, for instance, posting the content as images containing the text, rather than the text itself. Although optical character recognition can be employed to solve the particular problem, this further demonstrates the difficulty of hate speech detection going forward. It will be a constant battle between those trying to spread hateful content and those trying to block it.

Conclusion

As hate speech continues to be a societal problem, the need for automatic hate speech detection systems becomes more apparent. In this report, we proposed a solution to the detection of hate speech and offensive language on Twitter through machine learning using Bag of Words and TF IDF values. We performed comparative analysis of Logistic Regression, Naive Bayes, Decision Tree, Random Forest and Gradient Boosting on various sets of feature values and model parameters. The results showed that Logistic Regression performs comparatively better with the TF IDF approach.
We presented the current problems for this task and our system that achieves reasonable accuracy (89%) as well as recall (84%). Given all the challenges that remain, there is a need for more research on this problem, including both technical and practical matters.

References

[1] ncbi.nlm.nih.gov/pmc/articles/PMC6701757/ [Online]
[2] A. Gaydhani, V. Doma, S. Kendre and L. Bhagwat, “Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TF IDF based approach”.
[3] https://www.nytimes.com/2019/11/12/us/hate-crimes-fbi-report.html