Detecting Hate Speech On Twitter

Principle of Data Mining CAP 5771

Gorav Gupta

Giancarlo Bambaren

Table of Contents

[Abstract 3](#_Toc531971912)

[Python program and Input/Output data files Location: 3](#_Toc531971913)

[Introduction 5](#_Toc531971914)

[Methodology 6](#_Toc531971915)

[Collecting Data 6](#_Toc531971916)

[Pre-processing Data 8](#_Toc531971917)

[Data Efficiency: 8](#_Toc531971918)

[Feature Extraction: 8](#_Toc531971919)

[Building and Testing the Classifiers 9](#_Toc531971920)

[Naïve Bayes - First Execution: 9](#_Toc531971921)

[Naïve Bayes -Second Execution 9](#_Toc531971922)

[Max Entropy classifier - First Execution: 9](#_Toc531971923)

[Max Entropy classifier - Second Execution: 10](#_Toc531971924)

[Visualization of results: 10](#_Toc531971925)

[Limitations and further Improvements 11](#_Toc531971926)

[References 12](#_Toc531971927)

[Addendum 12](#_Toc531971928)

# Abstract

With the growth of social media becoming an extension of oneself and representing their values as well as their employers, the detection of potential hate speech in a person’s social media account has gained considerable interest from companies, especially in the hiring process. A successful supervised classifier should expedite the search for hateful content in a person’s social media account such as Twitter, rather than do it manually. In theory, this should be a simple task of searching for terms contained inside a hate speech lexicon in a person’s social media account. However, the scope of this task goes beyond that as hate speech deals more with a person’s sentiment.

With this in mind, it’s important to chose the right classifiers for this task. Naïve Bayes, Support Vector Machines, Decision Trees and Max Entropy all being attractive options, as they work well with text. Using Naïve Bayes and Max Entropy in this case, we sought to successfully classify tweets as “hate” or “no\_hate”. Naïve Bayes is a simple algorithm in comparison to the rest of the classifiers. It handles text data well. It is also not as resource intensive as the rest, which allowed us more leeway when it came to testing it and making changes, especially when working with data sets ranging from 1000 to 160,000 texts. Max Entropy is a lot more sophisticated than Naïve Bayes and can possibly give you a better fitted model. It also makes no assumptions about the conditional independence between terms/words, unlike the Naïve Bayes model. However, due to the model needing to optimize itself to estimate the parameters of the model, it is very resource intensive, so it requires more processing power, as well as time.

By training and retraining these models, we have shown that these classifiers can detect tweets as hateful or not hateful with a certain degree of accuracy. In particular, due to Naïve Bayes’ much faster training and testing time, it can be seen as the most efficient out of the two models when it comes to solving this problem with certain hardware limitations.

# Python program and Input/Output data files Location:

This is the first link for Max Entropy Files:

<https://www.dropbox.com/sh/rc2b315938qe5lw/AABQU3WQQHBqTVbrQ9EtG5ona?dl=0>

This is the second link for Naive Bayes files:

<https://www.dropbox.com/sh/229d6p9gask3z40/AADATUIXeDFAJvwic7LkmQmVa?dl=0>

# Introduction

Background and Motivation

In today’s world, social media has become an extension of oneself. Everything a person posts can be taken as a representation of who that person is. This is especially important for companies as their employees’ posts become a representation of the company and what it stands for. People’s posts can range from texts, pictures, audio, and video. The most common social media platforms include Facebook, Youtube, Twitter, Instagram, Snapchat, and Tumblr. For this project we decided to take a look at Twitter.

Twitter is primarily a text-based social media (although it is not rare to see these texts accompanied by a picture or video). Each user posts what are called “tweets”. Tweets vary in length up to an amount of 280 characters (double from what it was in early 2017). Along with pictures and videos, a person can also include emojis or hashtags to evoke certain emotions or sentiments. For very active users, Twitter only allows access up to the last 3,200 tweets that user has posted. The date of these tweets is timeless, so one could access a tweet that was posted a decade ago granted that user has not gone over 3,200 tweets.

So why twitter? Twitter, as mentioned is one of the most popular social media platforms with around 336,000,000 monthly active users. Various different people from all backgrounds use it on a daily basis. Not only that but it is also one of the most used social media platforms for companies, and provides various different ways of measuring customer satisfaction with promotions or new products. Twitter has a very solid and user friendly API through their developer program. With this tool, one can easily extract a particular user’s tweets in a particular time period of interest.

With this project we wish to create a hate speech classifier which can be used by human resources department of companies to examine potential employee’s twitter account. Because what an employee posts can be seen as a representation of their employer’s values, it is important to weed out any candidates that have posted hateful tweets. Manually going through a candidate’s potential thousands of tweets would be an extremely time consuming and painstaking process. A classifier would speed up the process drastically, allowing companies to review thousands of candidates throughout their decision making process.

# Methodology

Before starting building the classifier, we had to decide which type of classification we would use, whether supervised or unsupervised classification. Both methods have their strengths and weaknesses, but we decided supervised would be the best way as it is efficient to build a classifier from a labeled training set which provides us with a measurement of how well the classifier is working through the accuracy, rather than have no indication of the accuracy of an unsupervised classifier.

We also knew it was important to decide a way in which we would approach our classifier. We started our project using a hate speech lexicon and used this to classify tweets as hateful or non-hateful based on just one word from the lexicon, then we ran into the problem of potentially classifying non-hateful tweets as hateful because the context was not being taken into account. Using this method, we also run into a dimensionality problem which deems all supervised classifiers as unusable due to the single dimensional nature of the data.

In order to solve these approaches, we decided the best approach would be to build a hate speech classifier based on sentiment. Therefore we decided to take into account not only the hateful words, but also those words that describe the hateful sentiment within the tweet. A similar approach to sentiment analysis but this time only dealing with tweets being hateful or non-hateful.

# Collecting Data

When collecting the data, we knew because we were going to use a supervised classifier, it was important for us to have labeled data sets. The texts had to each convey a sentiment of either hate or no hate and had to have a corresponding class label in a separate column. We manually wrote or collected 1000 “tweets” and labeled them accordingly in a csv file. This set of data would then be split into the training and testing set. We ended up with 254 hateful tweets, and the remaining non hateful.

However due to a lack of accuracy in our classifiers (this will be discussed in the next section), we decided that increasing the amount of training set samples would improve our accuracy. Now doing this manually would have taken a very long time, so we had to search for an online dataset. Luckily Jigsaw/Conversation AI had a testing set consisting of 160,000 samples on Kaggle. The only problem we were facing with this data set was that the content was not labeled as hateful or non hateful, but instead had 6 different categories: toxic, severe\_toxic, obscene, threat, insult, identity\_hate. Because all of these can be considered as hateful, we decided that if a text was labeled as one of these 6 categories or more, then it would be considered as hateful. This resulted in 15,000 out of the 160,000 texts being hateful.

# Pre-processing Data

Before building and testing the classifiers, it was first important to extract our features for our classifiers from out data, as well as manipulate the text if necessary. For this we made user-defined functions that would be used in pre-processing this data. Two type of functions used are “Data Efficiency” and “Feature Extraction”

*replaceTwoOrMore*, *getStopWordList*, *getFeatureVector*, and *extract\_features*.

## Data Efficiency:

Under this step, our aim was to remove all the non-sentimental text from tweets so that our model can be executed faster and avoid creating redundant rules. So we used ***processTweet*** which is handling all of the symbols and text discrepancies that could possibly occur on a tweet. This function is important because not everyone tweets the same way and with proper grammar. In this tweets, all of the letters were converted into lowercase, all of the ‘[www.](http://www.)’ and ‘https’ were converted into a URL, the ‘@username’ was converted into AT\_USER, all additional whitespaces were removed, and hashtags were removed keeping the word after the #.

## Feature Extraction:

Under this step, our aim was to extract the features required for our Naïve Bayes & Max Entropy based models. So we used ***replaceTwoOrMore*, *getStopWordList*, *getFeatureVector*, and *extract\_features***.

The *replaceTwoOrMore* function replaces two or more repetitions of a character in a word with the character itself. The third function *getStopWordList*, reads a stopword text file and builds a list. The last two functions deal with the feature extractions and were vital for the analysis of the texts. The function *getFeatureVector* would split the tweet into words. Then for each word in the tweet, the *replaceTwoOrMore* function would be applied to them, the punctuation would be stripped, then the word was checked to see if it would start with a letter from the English alphabet (because we were focusing on the English language), and lastly it would be appended to an initially empty feature vector in lower case. Lastly, the *extract\_features* function would extract the words with the highest occurrence and most important ones from the tweets in the feature list and indicate whether they were important for hateful or non hateful tweets.

These functions would also be applied to the tweets that would be extracted from Twitter using Tweepy so that the tweets were formatted in the same format as the feature sets and would make it easier for the classifier to classify them.

Lastly we also obtained an **English hate speech lexicon from hatebase.org** which we used to pre-filter the tweets from a user which contained words that were in the lexicon. These tweets would then be the only ones to run through our classifiers as it would help save time for the classification.

# Building and Testing the Classifiers

Our first stage when building the classifiers was to decide which classifiers we would use for our objective. The classifiers Naïve Bayes, Support Vector Machine, Max Entropy and Decision Trees were all very attractive solutions. However we decided to go with Naïve Bayes and Max Entropy. Now why these classifiers? Naïve Bayes was a simple choice because it is a simple algorithm in comparison to the rest of the classifiers. It handles text data well. It is also not as resource intensive as the rest, which allowed us more leeway when it came to testing it and making changes, especially when eventually working with 160,000 texts. It is trained very quickly but unfortunately is not as sophisticated as the rest of them which can cause the accuracy to be lower than it could potentially be. This is also why we decided to go with Max Entropy. Max Entropy is a lot more sophisticated than Naïve Bayes and therefore can give you a better fitted model. It also makes no assumptions about the conditional independence between terms/words, unlike the Naïve Bayes model. The downside is that it is very resource intensive, so it requires more processing power, as well as time. This is because Max Entropy needs to optimize itself to estimate the parameters of the model it is making.

## Naïve Bayes - First Execution:

When training the Naïve Bayes model we first started with the 1000 tweets data set (split 800 into a training set and 200 into a testing set). The training of the classifier was done very quickly (around 10 seconds) and resulted in an accuracy of 0.955! This was very surprising so our first assumption was that the model was overfitting, and as it turns out, it didn’t do so well when it came to testing a few of our tweets from our twitter account. For example “I really dislike n\*\*s, wish they did not exist” was misclassified as no\_hate. Because the Naïve Bayes algorithm does not have many variables that can be manipulated we decided it would be best to retrain the model with a larger data set.

## Naïve Bayes -Second Execution

For this execution, we decided to use the 160,000 texts as our training set, and use the 1000 original tweet data set as our testing set. Training this model was a lot more time consuming (took 12 hours to train rather than 10 seconds) but did not give any memory issues. After this model was trained, the accuracy came out to 0.89. Although the accuracy was lower than the previous version of the classifier, the application seems to be slightly better than the previous one. This time, it successfully classified “I really dislike n\*\*s, wish they did not exist” as hate, along with other ‘tweets’ that had been misclassified.

## Max Entropy classifier - First Execution:

For our first version of the Max Entropy classifier, we chose the ‘GIS’ algorithm, and chose to run 10 iterations of the classifier with the 1000 texts training set. This is because usually at 10 iterations, the accuracy tends to reach a plateau. However, ever since the first iteration, the Max Entropy classifier’s accuracy remained equal, which was 0.746. This was also after training the model for 20 minutes. This level of accuracy is not bad but it left more to be desired considering it is a more sophisticated system than Naïve Bayes. This is especially important because it misclassified the previous tweet “I really dislike n\*\*s, wish they did not exist” as no\_hate. We also got the same level of accuracy when replacing the ‘GIS’ algorithm with the ‘IIS’ algorithm, both of which handle text classification well. We then thought that the problem could lie with our training set being too low, so we decided it would be best to retrain the model with larger data set.

## Max Entropy classifier - Second Execution:

For this execution, we decided to use the 160,000 texts as our training set, and use the 1000 original tweet data set as our testing set. Unfortunately after a couple of hours we got a memory issue as the memory was not sufficient. We then lowered the set to 30,000, and got the same issue after another couple of hours. This continued all the way to 10,000 sample texts, and unfortunately were not able to train the classifier with a high quantity.

Visualization of results:

# Limitations and further Improvements

There are plenty of improvements that could have been done in order to get a higher level of accuracy in our classifiers but first it is important to look at the limitations of the project.

* One of the biggest limitations was the time-frame for a project of this magnitude. Unfortunately making the 1000 tweets data set took a bit of time (approximately 12 hours as different cases were taken into account) which is why we had to work with a larger dataset that was not designed by us in order to try to improve the classifiers.
* Training the classifiers also took a lot of time, especially the Max Entropy classifier due to the complexity of the algorithm.
* The other largest limitation we had was our hardware. As explained, we were unable to train the Max Entropy classifier using the 160,000 training set due to our memory issues while running the iterations. Better hardware could have allowed us to not only be able to train the Max Entropy classifier with the 160,000 training set but could have also allowed us to train and test these classifiers at much higher speeds, saving us valuable time.

Now to talk about the improvements that could have been done it is first important to talk about the way in which we approached our classifier training. Perhaps cross-validation of the 1000 texts data set could have worked well with the classifiers and helped improve their accuracy, rather than just run the classifiers through the 1000 texts data set. Another way we could have improved the accuracy and efficiency of the classifiers could have been to include a function that looked at pairs of words rather than single words for the feature extractions. A lot of sentiment can be read when words are paired rather than looked at independently as pairs of words tend to show a lot more context rather than a single word. Other classifiers than Naïve Bayes and Max Entropy could have also worked well or even been better such as SVM, however we did not get the chance to get to this. A fourth improvement that could have made the project a bit more successful (if not by much) could be designing a function that corrects tweets in which numbers are being used as letters. Currently this is a way in which, not only our classifier, but others can be bypassed as this is a complicated problem to solve.

Lastly the classifiers have their own limitations as of now. For example, they provide no analysis of the visuals a person posts on their twitter account, we are only basing if a person has hateful content in their Twitter page based on their textual tweets. There is also no analysis of the retweets or likes that a user makes on their profile, both of which are good indication of the morals of a person. It is also limited to the English language, which limits the analysis on only English tweeting users.

# References

<https://www.nltk.org/_modules/nltk/classify/naivebayes.html>

<https://www.tutorialspoint.com/python3/python_reg_expressions.htm>

<http://billchambers.me/tutorials/2015/01/14/python-nlp-cheatsheet-nltk-scikit-learn.html>

<https://www.ravikiranj.net/posts/2012/code/how-build-twitter-sentiment-analyzer/#positive-tweets>

<https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data>

<https://gist.github.com/yanofsky/5436496>

# Addendum

Program Files

* First NBClassifier.ipynb
  + First executable program of Naïve Bayes: It is tagging tweets, solely based upon whether it has a hate word or not i.e. ignoring the sentiments
* Second NBClassifier 160k.ipynb
  + It is tagging tweets based upon both hate words as well as sentiments
* First Max Entropy Classifier.ipynb
  + It is tagging tweets as "hate" or "no\_hate" while not assuming that the features in the feature set are conditionally independent of one another.
* Hate\_Lexicon.csv
  + List of potential hate words, having those words makes tweet a potential "no\_hate"
* stopwords.txt
  + List of stop words which don’t reflect the sentiment of tweets
* sampleTweets.csv
  + Input file having 160K tweets with corresponding tags
* TestingTweets.csv *(For Naïve Bayes)*
  + First 1000 entires of orginal file with 160k tweets - Used for cross-validation
* TestingTweets\_GG.csv
  + Manually created/collected list of 1000 tweets with sentimenal value of "hate" or "no\_hate"
* TestingTweets200.csv
  + Last 200 out of 1000 tweets from manually created/collected list to test the accuracy of the first Naïve Bayes classifier
* finalpreserve
  + Dump of Trained Naïve Bayers model using joblib.dump
* Hate speech training and testing.csv
  + Manually created/collected list of 1000 tweets with sentimenal value of "hate" or "no\_hate"
* Hate speech training and testing 800.csv
  + First 800 out of 1000 tweets from manually created/collected list to train the first Naïve Bayes classifier
* 10sampleTweets.csv
  + List of 10 tweets used to test the tweets