Sentiment Analysis of Twitter Data

**Dean D’souza, Student, Harrisburg University of Science and Technology,** [**dpdsouza@my.harrisburgu.edu**](mailto:dpdsouza@my.harrisburgu.edu)

**Lecturer: Majid Shaalan, PhD., Associate Professor of Computer Science, Harrisburg University of Science and Technology,** [**MShaalan@Harrisburgu.edu**](mailto:MShaalan@Harrisburgu.edu)

**Assistant Lecturer: Raja Shanmugavelan, Lecturer of Computer Information Systems,** [**RShanmugavelan@Harrisburgu.edu**](mailto:RShanmugavelan@Harrisburgu.edu)

# Abstract:

Sentiment analysis is a process which makes use of various methods such as Natural Language Processing, Text Analysis, etc. to find and extract information from textual data. [1] It can be applied to various kinds of text data, out of which a popular source is from twitter, a social networking website. The project aims at replicating and improving on certain steps in analyzing twitter data effectively through the python package of nltk, allowing for an easier and simple implementation.

**Keywords:** Sentiment Analysis, Natural Language Processing, Text Analysis, Twitter, nltk

# Introduction:

Sentiment Analysis (also known as opinion mining) is an important field in Computer Science which focuses on obtaining information from text data, usually in the form of attitude of the author with regards to a topic or the overall polarity of the document. It involves a variety techniques to achieve this goal, which usually involves Natural Language Processing, Text Analysis, and Computational Linguistics.

Natural Language Processing (or NLP) is another important field in Computer Science (related to Human Computer Interaction) which focuses on how computers and humans interact through human language. NLP generally utilizes Machine Learning Techniques and systems built in such a fashion have the advantages of having automatic learning procedures which focuses on the most common cases, makes use of statistical inference algorithms, and having the ability of improving accuracy through the addition of more input data.

NLP itself consists of several tasks, one of which is sentiment analysis of text data, and involves some of the following steps:

1. Tokenization: which is the process of breaking down a collection of characters or a string into smaller parts known as tokens. In terms of breaking down textual data, this would involve breaking down the data into individual words, punctuations, and possible numerical data.
2. Chunking: which involves breaking down a collection of strings into individual strings. In more general terms this can be thought of as breaking down paragraphs into individual sentences, or even breaking down long sentences into individual units.
3. Stemming: which involves converting word tokens into their base form. An easier way to understand this would be with the example of the word “looking”, which is one of the many forms of the word “look”. Obtaining this base form of “look” is known as stemming.
4. Part-of-Speech tagging: which is an important task on its own and involves correctly identifying what part of speech each word token belongs to. An example of this would be the word token “book” which could be a noun (“The book on the shelf”) or a verb (“I booked a flight to Las Vegas”).

Many of the above-mentioned tasks can be easily performed in python using the Natural Language Tool Kit (NLTK), which provides many pre-packaged functions and data structures. NLTK in one of the most popular and easy to use toolkits for performing Sentiment Analysis through python. Some of the important tools that are used in this project include the Naïve Bayes Classifier and the Decision Tree Classifier algorithms.

The Project makes subtle improvements on the tasks of tokenization and part-of-speech tagging to obtain required features for building a model which can automatically tag tweets (twitter text or posts) as positive or negative.

# Data:

The original dataset was obtained from the Sentiment140 website [3] and consists of around 1,600,000 records (tweets) collected over a period of a few days. The original authors utilized the Twitter Search API to collect tweets by using a keyword search and then processed the data by matching emoticons to determine the polarity of the tweet, i.e., tweets with emoticons like ‘☺’ were labelled as positive while those with emoticons like ‘☹’ were labelled as negative. The data file which is available for download is in the form of a ‘.csv’ file and has the following fields [3]:

1. The polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
2. The id of the tweet (2087)
3. The date of the tweet (Sat May 16 23:58:44 UTC 2009)
4. The Query. If there is no query, then this value is NO\_QUERY.
5. The user that tweeted (eg. robotickilldozr)
6. The text of the tweet (eg. Lyx is cool)

As there were a few Hardware limitations which prevented the use of the entire data, a sample of the original was taken for the project. This sample consisted of a total of 7200 records, out of which 3600 were negative (or 0) and 3600 were positive (or 4). Once again, due to Hardware limitations, samples of the type neutral (or 2) were not taken. However, some previous work on the same recommends that at least 600 records of each type is present for training the models and hence the current sample is sufficient. [5]

To process the data efficiently, a few fields had to be dropped, namely:

1. Tweet ID: this field was dropped as it was a unique number which identified each tweet and did not contribute much towards estimating the tweet polarity.
2. Date of the Tweet: while this field may have contributed towards the polarity of the tweet, the sample extracted was within a timespan of less than a day (between late Monday and early Tuesday) and hence had no significant effect on tweet polarity.
3. The Query: several records showed the value NO\_QUERY, especially in the sample. Hence, this field was removed as it would not contribute much to the classification task.
4. The User / Author: the sample contained a variety of authors, and for the most part, each record displayed unique authors. Hence, it was removed from consideration.

The variables or fields left were the tweet polarity and the actual contents of the tweet, which under most circumstances is more than enough for classification of twitter data.

The characteristics of the sample tweets can be best seen through the following statistics:



Figure Lexical Diversity

As we can see from the above screen shot, the lexical diversity, which is a measure of the range of the vocabulary, is 5.92 or roughly 6, which is on the lower end of what would be considered good for Sentiment Analysis.

We now try to see which tokens or word types are used the most often.

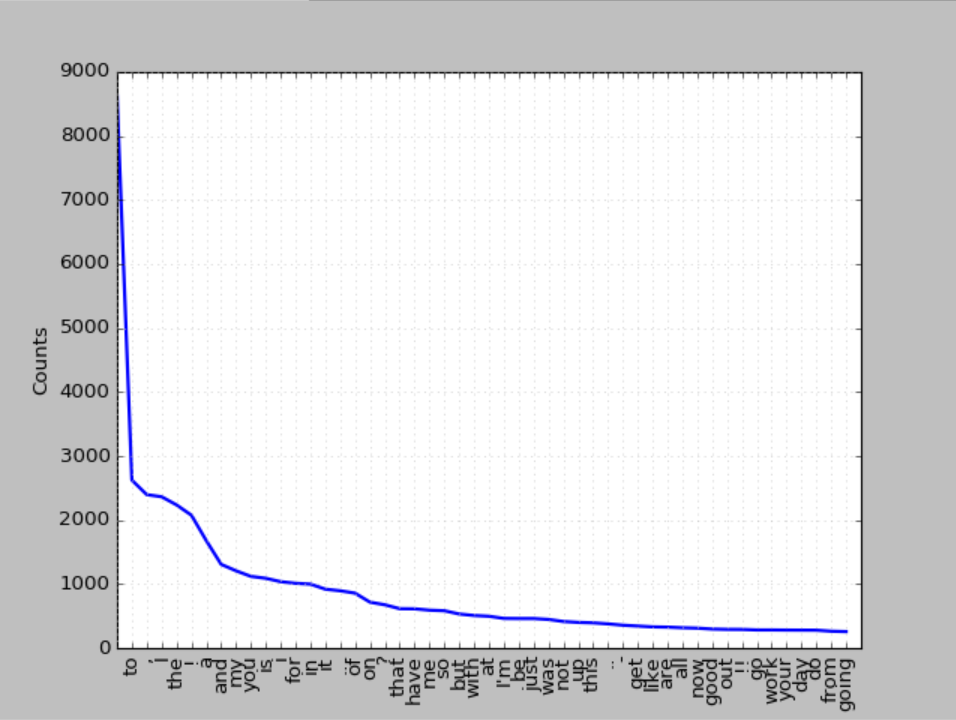


Figure Frequency Distribution of Top 50 Tokens

As we can see above, tokenizing the text seems to have introduced an empty token which has the most frequency in the text. However, if we ignore this observation we can see that the most used word or token is ‘to’ followed by ‘,’ and so on.

We now look at some of the interesting collocations that are contained in the data as follows:

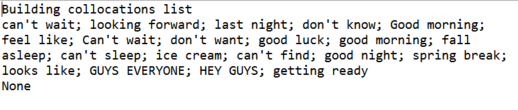


Figure Collocations

Collocations, which are the most frequently occurring pairs of words that are considered to be too frequent to be by chance, can improve our understanding of the data and also help in building a more accurate model.

We now look at the important parts-of-speech that are contained in the data:

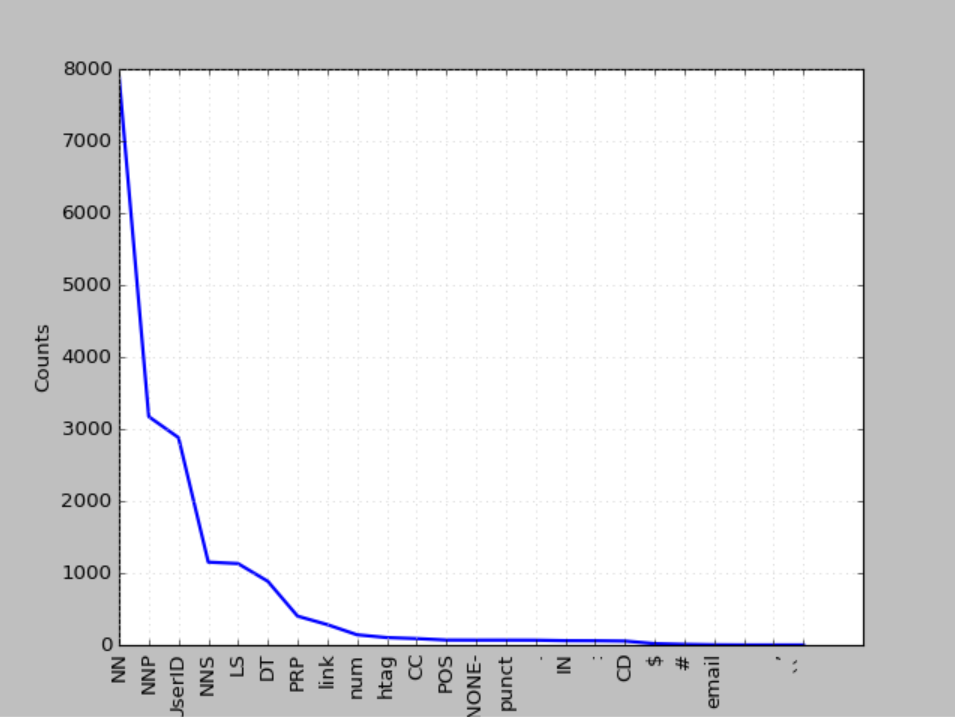


Figure Frequncy Distribution plot of Parts-of-Speech

As we can see from the above frequency distribution plot, several of the words contained in the data belong to the ‘NN’ or Noun type, followed by ‘NNP’ or Proper Noun and so on. We can also see that there is a flaw in the POS tagger which fails to tag certain punctuations and fails to tag a few tokens completely as seen by the ‘-NONE-’ tag.

As this issue, could not be rectified even with the use of the default tagger which comes pre-packaged with NLTK, we leave them as is.

# The Model:

While the original authors built a Maximum Entropy Classifier model for classifying the data, the Maximum Entropy Classifier supplied with NLTK proves to be quite memory intensive which would often lead to memory errors on machines with less RAM or with older processors. Hence, the project utilizes two classifiers to achieve similar results.

The first classifier is the Naïve Bayes Classifier which comes pre-packaged with NLTK. The naïve Bayes Classifier is based on Bayes theorem which is based on the assumption that all features are independent of each other. This also means that every feature gets a say in determining the classification of the given data. In NLTK, the Naïve Bayes Classifier begins by calculating the prior probability of each label (determined by checking the frequency of each label in the training set) which is then combined with the contribution from each feature to perform the classification. [4]

The second classifier is the Decision Tree Classifier which also comes pre-packaged with NLTK. The algorithm for building the classifier works on the idea of selecting the best decision stump based on all the available features and then making incremental improvements by removing unnecessary features. [4]

The features used in the building the classifiers is whether a tweet contains a word (if the word is at least 3 characters, this is mainly done to eliminate most of the stop words that occur in text) and its overall frequency of occurrence.

We now begin to construct the required models as follows:

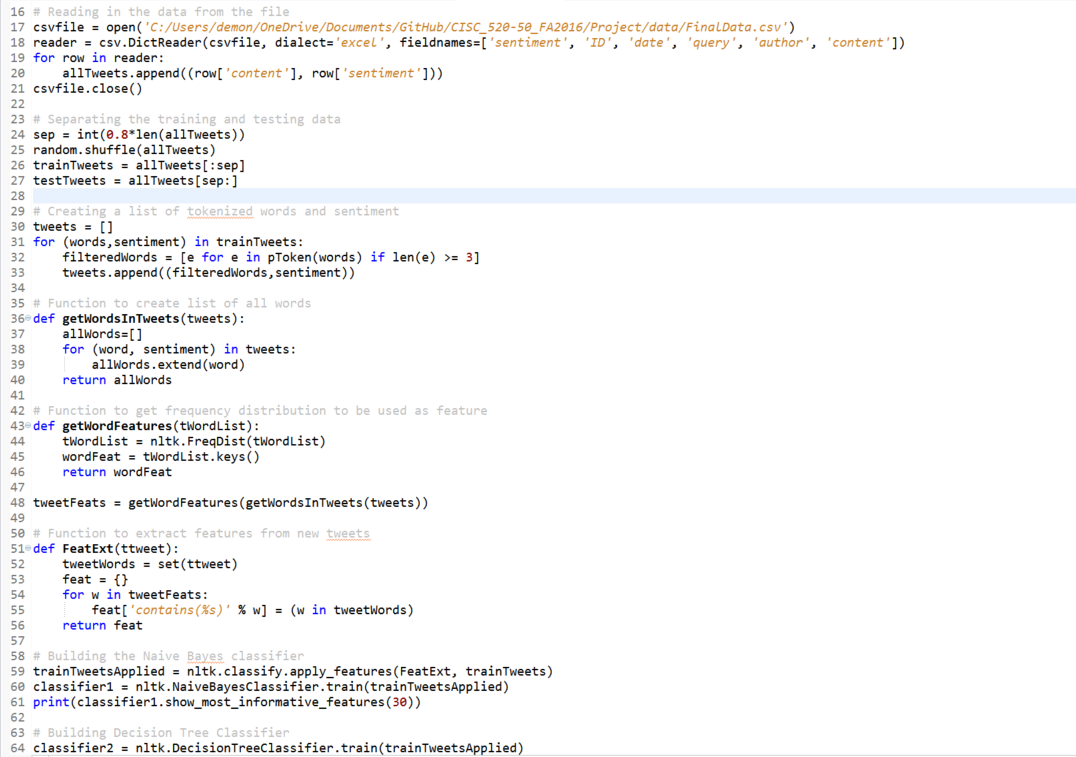


Figure Code Snippet showing how the classifiers are built

# Evaluation:

Evaluation of the model was done using 20% of the data which included equal proportions of positive and negative tweets.

With regards to the Naïve Bayes Classifier, we can see the 30 most informative features as follows:

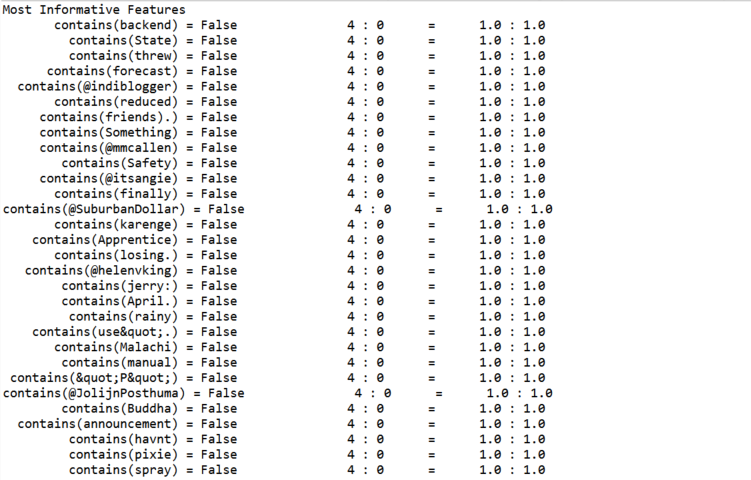


Figure Most Informative Features of the Naive Bayes Model

The evaluation of the models is done as follows:

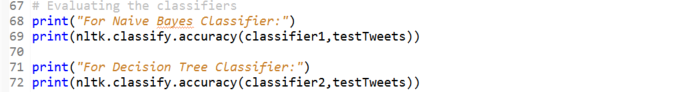


Figure Code Snippet of evaluating Accuracy

\*\* Due to errors in incorporating POS tag features and lengthy runtime (and crashes of the program) result of accuracy is not included. Results will be included in final white paper and presentation. \*\*

# Conclusion:

While the models seems to achieve decent accuracy in classifying the data, several improvements can still be made.

## Future work:

The model shows much need for improvement and areas of improvement are as follows:

1. The process of tokenization could be improved by including conditions to remove periods that occur at the end of sentences. Another issue was handling the occurrence of ‘&quot;’, a result of improper reading of the csv files content field, which could be improved by adding additional possible variations of its occurrence. Further work can also be done to obtain the stem of words which can improve the feature sets.
2. The part-of-speech tagger is not always accurate especially when it comes to distinguishing between links of the type ‘g2a.com’ and improperly tokenized sentence endings. There also seems to be a problem in detecting periods and classifying them as punctuations.
3. Additional data is needed to further improve the accuracy of the classifiers. However, a balance needs to be obtained to prevent crashes on older systems.
4. Feature sets could be improved to increase the accuracy of the classifiers. Inclusion of Part-of-Speech feature needs to be incorporated.
5. Rather than just classifying the text to be only positive or negative, additional classifications could be made to give more value to each tweet.

# References:

[1] “Sentiment Analysis”, Wikipedia, (<https://en.wikipedia.org/wiki/Sentiment_analysis>)

[2] “Natural Language Processing”, Wikipedia, (<https://en.wikipedia.org/wiki/Natural_language_processing>)

[3] “Sentiment140 – For Academics”, Sentiment140, (<http://help.sentiment140.com/for-students>)

[4] “Natural Language Processing with Python”, Steven Bird, Ewan Klein, and Edward Loper, O'Reilly Media, 2009, (<http://www.nltk.org/book_1ed/>)

[5] “Twitter Sentiment Analysis using Python and NLTK”, Laurent Luce, January 2nd 2012, (<http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/>)