**SPAM DETECTION ON TWITTER**

**Project as part of CS 63001.004: Analysing & Securing Social Media**

Aviral Verma (axv190007)

Jitendra Justin (jxj172430)

Kaustubh Deshpande (kxd180005)

1. **INTRODUCTION**

Twitter gives users an opportunity to share their messages concerning everything including news, occasions, celebrities, political issues, and so on. Compared to various other social media platforms, the connection between users is bi-directional rather than unidirectional connection. This means that a particular user may not be following one of his followers. Twitter was overwhelmed by a lot of malignant tweets that were sent by a huge number of spammed users account; this occurred around April of 2014. The authentic users mistake the spam information for as important one. Spam messages are difficult to regulate. Email administrations such as Gmail, Microsoft etc. are still finding better ways to prevent spam.

Twitter utilises both manual and mechanised administrations to fight spammers. The manual method involves Twitter giving users a chance to report spammers via the spammers' profile pages. These manual methodologies are stressful and may not be sufficient to distinguish between all spammers because of billions of users.

The possession of a huge number of spam protests documented against the account, the following/unfollowing of a huge number of accounts within a brief span, the publishing of a copy of the messages in a single account, the publishing of malignant connections are the ways in which Twitter aims to handle the spam accounts and tweets.

The users are more concerned with spam accounts and tweets on Twitter because there is a high increase in spam account in the Twitter platform since its inception. Average number of spam reports has been around 25,000 a day in 2017 to 17,000 a day in 2018.

1. **SUGGESTED MODEL**

Twitter API tweepy is used to get tweets from the website. In spam accounts, features like followers\_count,favourites\_count,  profile\_image, friends\_count, geo\_enabled, time\_zone,location, will be very low. At the same time, features like total\_hashtags and total\_links will be very high for spam accounts and spam tweets. timeline = api.user\_timeline(screen\_name = user.screen\_name, count = 100, include\_rts = True) is the code to get information from Twitter. Here, API used is the tweepy API.

Twitter considers an account as spam “if a user posts multiple unrelated updates to a topic using the #symbol.” We count the number of hashtags in the 100 most recent tweets of the user, and this feature is used to classify as spam or real.

Data pre-processing in our model is achieved by the processes of Snowballing and TF-IDF Vectorization.

**MULTINOMIAL NAÏVE BAYES CLASSIFIER**

For the purpose of classification, we use the Multinomial Naïve Bayes classifier. Multinomial Naive Bayes classifier is a specific instance of a Naive Bayes classifier which uses a multinomial distribution for each of the features.

This works well for data which can easily be turned into counts, such as word counts in text. This is especially suitable for our data. This is a type of classifier which works on tokens, with spam or ham, in the tweets. It then uses Bayes theorem to calculate the probability that a particular tweet is spam or not. The technique can classify almost any sort of data.

The Dataset used in our model has been created through the Twitter API. The API was used to retrieve information regarding the Tweets and their posting accounts. Since images cannot be processed, we instead use the links these images direct to, contained in the HTTP header of the image. The Tweet texts were concatenated with these links for more efficient spam detection mechanism.

**CHALLENGES:**

* Difficulty in gathering labelled dataset in a limited time.
* Learning the required NLP (Natural Language Processing) concepts used in the project.
* Understanding Machine Learning concepts and Naive Bayes Algorithm from scratch, since most of us had not taken any related courses till now.

**ARCHITECTURE DIAGRAM**

**DATA**

Raw Data

Snowballing **PRE-PROCESSING** TF-IDF Vectorization

Pre-processed data

**FEATURE EXTRACTION**

Features

**MODEL TRAINING & TESTING**

**SAVED MODEL**

Live Tweet

**TWEET CAPTURING**

**SPAM**

**?**

**CLASSIFICATION**

HAM

1. **WORKING**

**Step 1: Data Collection and Categorization**

Accounts that have verified attribute as TRUE are real accounts. Tweets which do not have profile picture, background image, followers but use many links in the tweets and follow many accounts are considered spam accounts. Tweets which use many links, external links, shortened URLs, defamatory abusive language can be classified as spam. We also performed manual checking of the rest of the tweets and accounts in order to classify as spam or real from the Twitter data.

**Step 2: Data Pre-processing**

In order to increase the efficiency of our mechanism, data pre-processing is a necessary function. Data pre-processing involves the processes of Snowballing and TF-IDF Vectorization.

**Snowballing** consists of the following actions,

* Separate the sentence into individual words
* Convert all letters to lowercase
* Remove stop words
* Don't care for non-English words

**TF-IDF** stands for term frequency-inverse document frequency.

The TF-IDF weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus.

The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

TF: Term Frequency, which measures how frequently a term occurs in a document. TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)

IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. IDF(t) = log(Total number of documents / Number of documents with term t in it)

**Visualization:** Using the occurrence weights obtained by TF\_IDF Vectorization, we present a Word Cloud Visualization for spam and ham data. A word cloud is a novelty visual representation of text data, typically used to depict keyword metadata (tags) on websites, or to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or colour.

This format is useful for quickly perceiving the most prominent terms to determine its relative prominence. The word clouds generated for spam and ham data are provided in the Appendix A, Fig A.1 and A.2.

**Step 3: Feature Selection**

Favorite\_Count, retweet\_count, friends\_count, total\_mentions, total\_hashtags, total\_links and sample\_tweet. The features that can individually categorise a tweet  spam or ham are retweet\_count, total\_mentions, total\_links and total\_hashtags.

**Step 4: Model Training and Testing**

We scramble dataset and separate data for training and testing purposes. We use 80% of the data as Training data and remaining 20% as Testing data. Scrambling is random. We perform initial fitting of parameters on the training set and transform to a particular set of examples.

We extract the top features under the categories features\_train, features\_test, labels\_train, labels\_test. The top features extracted are used for improving the performance of the model.

We tested the trained model with the test data to obtain accuracy and F1 scores using Multinomial Naive Bayes. The trained model is saved to be further used in real time spam detection in a tweet.

**Step 5: Real Time Tweet Catching and Classification**

In order to catch a tweet in real time, the Tweepy API is used. A user must provide their Twitter Developer account credentials in the *credentials.py* file to capture a live tweet.

Using the model we generated by our MNB classifier, we classify the live tweet as spam or ham. The tweet captured is fed to the model and based on its features, it is classified by the model as Spam or Not Spam.

1. **RESULTS**

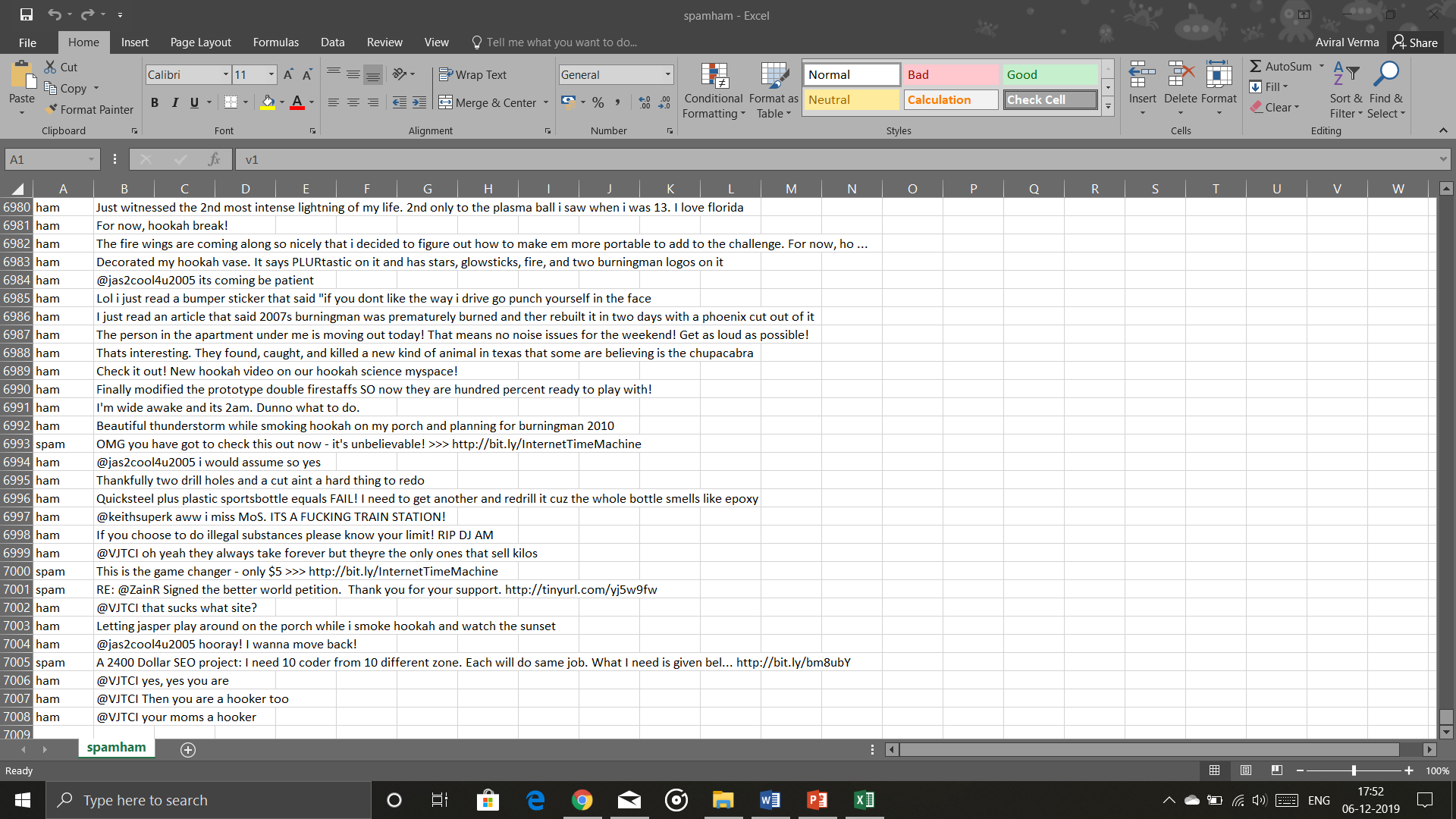
Running the *spam\_classifier.py* file with valid credentials in *credentials.py* provides us with results on a live tweet captured by *tweet\_catch.py*.

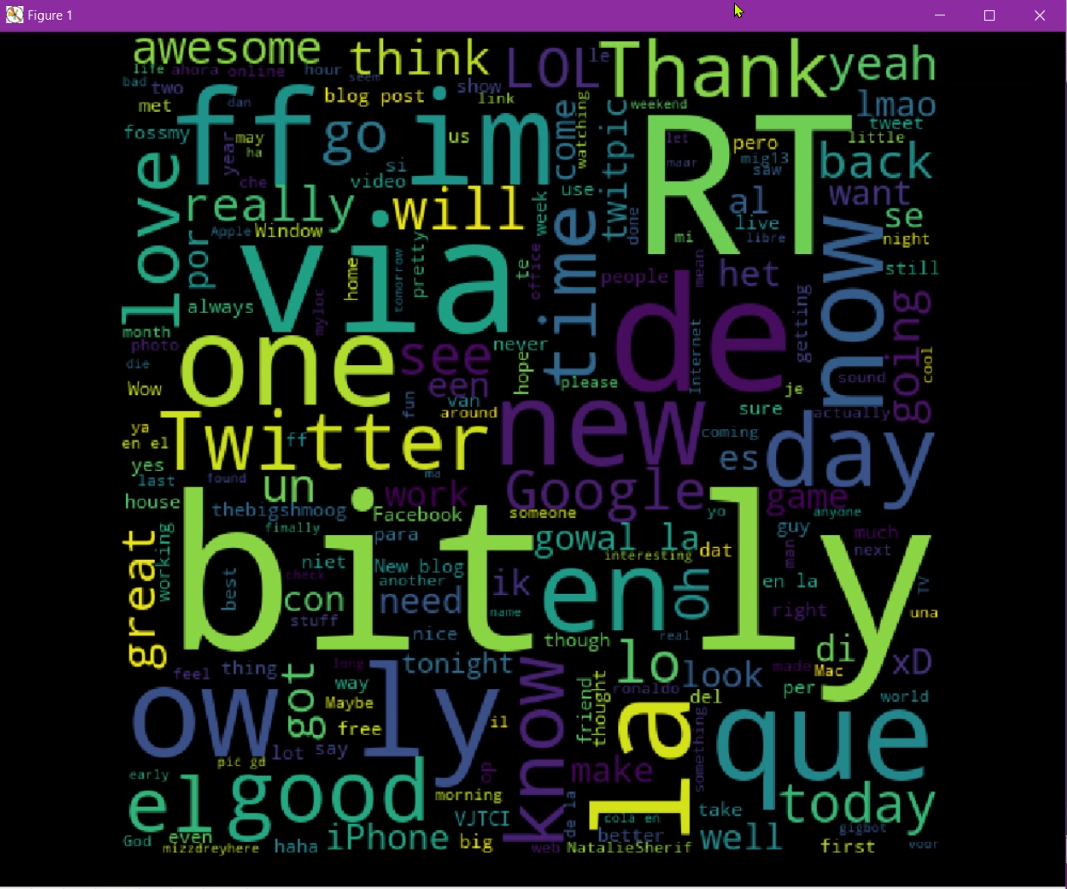
We were able to successfully classify a number of tweets as spam or ham.

After numerous iterations we found our model to display the following statistics,

|  |  |  |  |
| --- | --- | --- | --- |
| CLASSIFIER | ACCURACY | PRECISION | F1 SCORE |
| Bernoulli Naïve Bayes | 88.07 | 71.22 | 80.37 |
| Multinomial Naïve Bayes | 94.51 | 85.78 | 89.11 |

**APPENDIX A**

****Fig A.1 Dataset in .csv format

Fig A.2 HAM WORD CLOUD

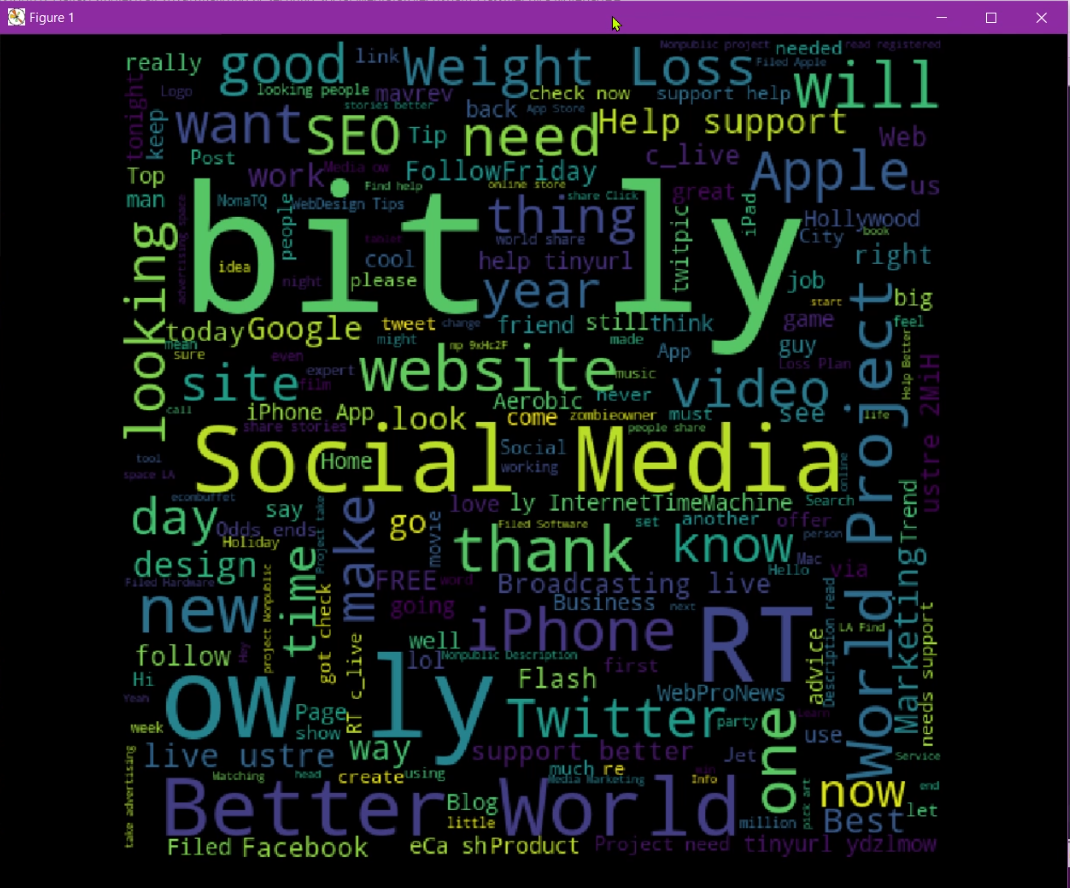


Fig A.3 SPAM WORD CLOUD

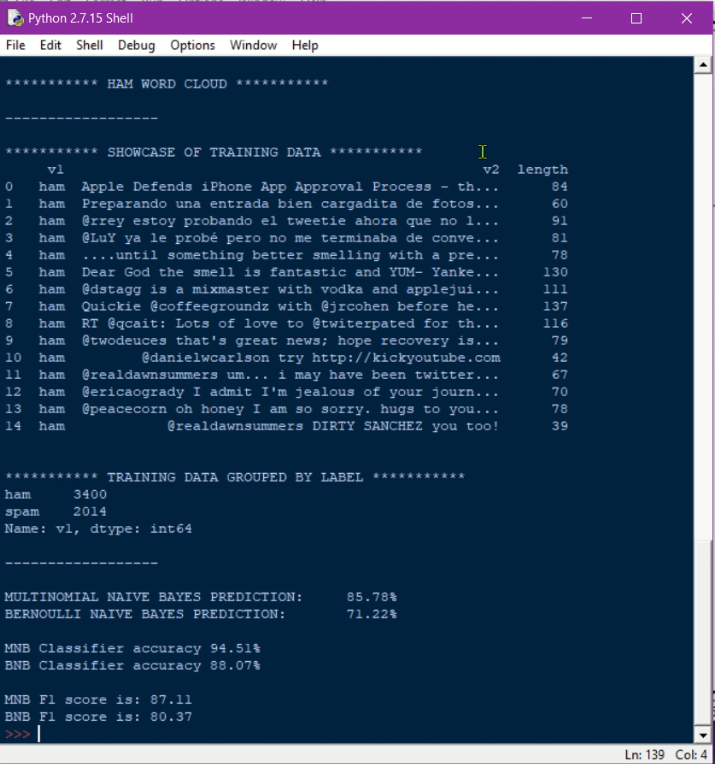


Fig A.4 CLASSIFICATION RESULTS

**Examples:**

Analysed Tweet:

“Starting today, you can now hide replies to your Tweets. Out of sight, out of mind “

According to Naive Bayes Classifier, this tweet is: HAM

Analysed Tweet:

“my friend told me about this site tinyurl.com/nukasflk, he got the fame and xbox completely free from there You can try It to and get the game for FREE! “

According to Naive Bayes Classifier, this tweet is: SPAM

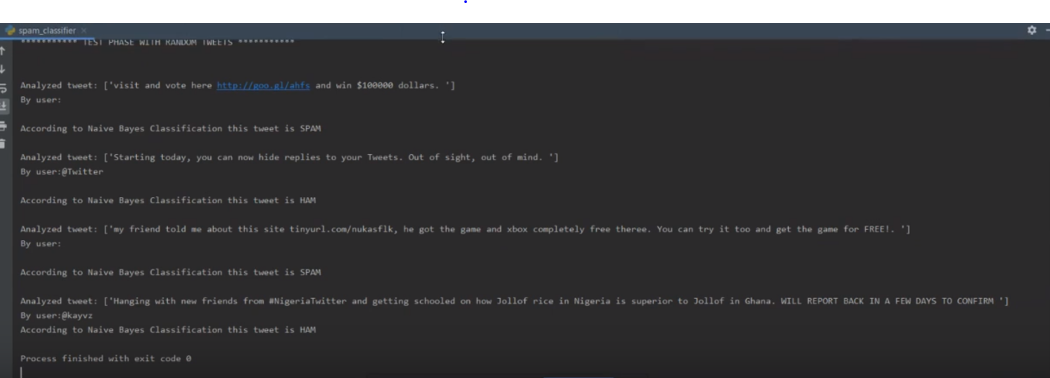


Fig A.5 Real Time Tweet Classification