Tweet Times and Vocabulary Sentiment of Donald J. Trump

Adrian Sandoval-Vargas

Department of Computer Science

Oakland University

**Abstract**

This study aims to figure out Donald J. Trump’s popular tweet times, topics in which he was active on, and his sentiment on spoken topics. The data was extracted using Twitter’s API across 16 days by searching the user @realdonaldtrump and filtered out any retweets. In this study we will look into data preprocessing, data visualization to extract key events, and perform sentiment analysis on the filtered data.

**Introduction**

This report will focus on how the data was extracted, cleaned, and processed. It will include visuals of the data collected and what tools were used to obtain the data and visualized. The data are tweets from Donald J. Trump from 11/20/2019 – 12/05/2019. These tweets were extracted from Twitter using Twitters API Calls. They were extracted in UTF-8 format to keep integrity of the data for processing the data. The data was filtered for https links, ‘@, #, &amp’. Stop words using the NLTK library, Visualizations were done by matplotlib, and Sentiment analysis was done using Textblob.

The paper will be laid out in 4 sections. In section 1, I will introduce the data, discuss and visualize key features, and perform statistical analysis to get a clean dataset. In section 2, I will discuss on random forest classifiers and my parameters as well as how to deal with data imbalance. Section 3 will be my experimentation and the tests conducted on the data. Section 4 will the final result and thoughts on what I should have done differently.

**1. Data Extraction, Data Cleansing, and Preprocessing**

The data was extracted from Twitter by using Tweepy’s library along with Twitter’s API keys. The extraction code:

auth = tweepy.OAuthHandler(key, sec)

auth.set\_access\_token(at, atc)

api = tweepy.API(auth,wait\_on\_rate\_limit=True)

with open('realdonaldtrump2.csv', 'a', encoding='utf-8', newline='') as file:

writer = csv.writer(file)

for tweet in tweepy.Cursor(api.user\_timeline,id='realdonaldtrump',

lang="en", include\_rts = False):

print (tweet.created\_at, tweet.text)

writer.writerow([tweet.created\_at, tweet.text])

This generated a csv file that we can now import into our project for data analysis. Once we import that data using pandas, we get a data frame that looks like so:

A screenshot of a cell phone

Description automatically generated

Figure 1: Raw data captured by Tweepy

As we can see we have many unfavorable text in our data. Our first step is to separate our Date column into Date and Time. To do simple I simply looped the dataframe[‘Date’] column and split the text by the whitespace in between. Since we won’t be looking into too much detail on the exact time he tweeted, we round the time to the hour.

A picture containing road

Description automatically generatedA screen shot of a computer

Description automatically generated

Figure 2: Hours data frame on the left, Date/Time on the right

Now that we have the time separated, we have to preprocess the text. First, we have to extract the urls from our tweets. This was done by using the UrlExtract Library. The code is:

w = []

for i in trump['text']:

if extractor.has\_urls(i):

url = extractor.find\_urls(i)

for k in url:

i = i.replace(k, '')

w.append(i)

else:

w.append(i)

This code checks to see there is a url in the tweet and replaces the url with whitespace. Next we have to remove the following symbols [‘@’, ‘#’, ‘&amp’]:

#remove '@' and '#'

for i in range(len(w)):

if '@' in w[i]:

w[i] = w[i].replace('@','')

if '#' in w[i]:

w[i] = w[i].replace('#','')

if '&amp' in w[i]:

w[i] = w[i].replace('&amp', '')

This generates an array of all the tweets with all the symbols not needed. Since we will be using the NLTK library for the stop words and the Sklearn library for the CountVectorizer function we need to make the ‘w’ array into a single array:

w = ' '.join(w)

z = []

z.append(w)

Now, to prepare the data for some analysis I will be making a boolean matrix with the following code:

x = vec.fit\_transform(z)

y =[i for i in vec.vocabulary\_.keys()]

y.reverse()

bM = pd.DataFrame(x.toarray(),index=['count'] ,columns=y).T

This puts our data at our disposal to further our knowledge on the data we mined.

**2. The Time Data**

In this section will dive into the timestamps of the tweets made by Donald Trump and point out key fetaures. Firstly, I would like to take a look at Trump’s overall tweets per hour

A screenshot of a cell phone

Description automatically generated

Figure 3: Trump’s Tweets per hour

As shown in figure 3, trump is super active during 12:00, 19:00, and 23:00. He averages 8 tweets per hour. Continuing,

A screenshot of a cell phone

Description automatically generated

Figure 4: Trump popular time of day to tweet

Trump seems to tweet more between 8pm – 6am as apposed to any other time of day.

Next I’ll be looking at the tweets per day information:

A screenshot of a cell phone

Description automatically generatedA close up of a piece of paper

Description automatically generated

Figure 5a: Trump’s tweets per day Figure 5b: Trump’s Time Vs Date tweets

As show in figure 5a, He had a significant twitter day on 2019-12-02. Also in figure 5b, we can see the relation bewteen the quantity of tweets and the time of day he is most active. Mornings are pretty dry and evenings are heavy with tweets.

**3. The Text Data**

To conduct all my tests I decided to use python as the langauge with common libraries suchas pandas, numpy, sklearn, matplotlib, scipy, etc. It is important to note that and where TP = True Positive results, FP = False Positive results, FN = False Negative and . Before continuing I would like to mention that for each experiment, the first test is without data imbalance techniques, hence the high percision.

**4. 12/02/2019 Case**

**5. Conclusion**

The result of this expirement varies on which key performance indicator you are willing to accept. For the purpose of obtaining the best accuracy, then experiment 3.5’s synthesized data results in 82.7% with a decent F1 and Recall percentage. If I had to conduct this study again, I would look into duplicate values and find a relation between the payment status (pay duly, 1 month over, etc), the bill amount, and how much the credit card holder paid.

**References**

[1] Original research paper on this dataset

*The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients –* <https://www.sciencedirect.com/science/article/pii/S0957417407006719>

[2] Credit Card Default Dataset <https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients>

[3] Random Forest Classification <https://www.kdnuggets.com/2017/10/random-forests-explained.html>

[4] Random Forest Classifier Python Documentation <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

[5] Synthetic Minority Over-Sampling Technique (SMOTE) <https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume16/chawla02a-html/chawla2002.html>