COMM 337- Project 1 Report
Group 39

Team Members: Kofi Buahin – 48959143 – kkbuahin@gmail.com

Data Collection Step Explanation:

The first task at hand was to collect all the necessary tweets to conduct the different analysis presented in the subsequent parts of Project 1. The two keywords I decided to use for this project were “Obama” and “Trump”. The first thing I did was to store the twitter credentials I had made in in a dictionary named “twit_cred” and then write them into a json file named 'Kofi_twitter_credentials.json' so that later on I could access them in order to access the twitter API and stream the necessary tweets. Then, once this was done, I read the json file of stored credentials to create a set of variables containing my consumer keys and access tokens to be called upon in later functions.

Then using code originally sourced from Ryan Mcgrath on Github, I was able to establish the class MyStreamer which included functions on_success, on_error and store_json. Taking in the four keys and access tokens as inputs, these functions served to ensure the received tweets were in English, each of the tweets collected was printed in a specific format in the spyder console, the code terminated when 10,000 tweets were collected (on_success) . They also served to disconnect in the case of an error (on_error) and store the tweets in the form of dictionaries in a json file once the code had finished (store_json).

```
# Github Code: https://github.com/ryanmcgrath/twython/blob/master/twython/streaming/api.py
class MyStreamer(TwythonStreamer):
    def on_success(self, data):
        if 'lang' in data and data['lang'] == 'en':
            tweets.append(data)
            print('received tweet #', len(tweets), data['text'][:100])
        if len(tweets) >= 10000:
            self.store_json()
            self.disconnect()
    def on_error(self, status_code, data):
        print(status_code, data)
        self.disconnect()
    def store_json(self):
        with open('tweet_stream_{}_{}.json'.format(keyword, len(tweets)), 'w') as f:
            json.dump(tweets, f, indent=4)

```

After doing this, I created a variable called stream which stored a value where I called the MyStreamer function using my 4 input variables as seen in the screen clipping below. Finally, I used the system arguments function to obtain the keyword that would be used to search for each of my keywords. I ran the code twice, each time altering the if statement below to change the keyword and collect the appropriate tweets.

```
if __name__ == '__main__':
    with open('Kofi_twitter_credentials.json', 'r') as f:
        credentials = json.load(f)
    KOFIC_KEY = credentials['CONSUMER_KEY']
    KOFIC_SECRET = credentials['CONSUMER_SECRET']
    KOFIA_TOKEN = credentials['ACCESS_TOKEN']
    KOFIA_TOKENSECRET = credentials['ACCESS_TOKEN_SECRET']
    stream = MyStreamer(KOFIC_KEY, KOFIC_SECRET, KOFIA_TOKEN, KOFIA_TOKENSECRET)
    if len(sys.argv) > 1:
        keyword = sys.argv[1]
    else:
        keyword = 'Obama'

    # Github Code: https://github.com/ryanmcgrath/twython/blob/master/twython/streaming/api.py
    # Accepted parameters: https://developer.twitter.com/en/docs/tweets/filter-realtime/api-reference/post-statuses-filter
    stream.statuses.filter(track=keyword)


```

Preliminary Analysis Step Explanation:

The next portion of project 1 was to conduct a preliminary analysis based on the tweets I had collected.

Sub Question A Explanation:

My first course of action was to import all the appropriate packages of functions that I would need for this analysis (json, counter, nltk , string, TextBlob matplotlib and WordCloud) . After doing this, I used **nltk.corpus.stopwords.words('english')** to create a variable containing a list of stored stopwords. I also made a variable called chars which contained certain characters that I wanted to remove from the text in my tweets. I wanted to ensure that I collected the appropriate text data from the dictionaries in the stored json that contained all the tweet data. To do this I used a combination of for loops to append a unique list (ex. obama_text) with the text values from the tweet data. Once I had the unique list of raw text from the collected tweets, I had to collect all the tweets in an area where I could edit them for stopwords and unnecessary characters. I created a new empty string variable (ex. obama_text2) to add each piece of stored tweet text to using a for loop. Once I had done this I created two lists, one for a list of all the words and one for the words excluding all the identified stop words. I did this by using the word_tokenize method from the nltk package. Once I had the two lists of words I used the nltk.FreqDist method to find the top 10 most common words for each keyword.

```
with open('tweet_stream_Obama_10000.json', 'r') as infile:
        storedtweets_Obama = json.load(infile)
obama_text = []
obama_text2 = ''
for item in storedtweets_Obama:
    obama_text.append(item['text'])
for tweet in obama_text:
    obama_text2 = obama_text2 + tweet + ' '
obama_words = nltk.word_tokenize(obama_text2)
for w in obama_words:
    if w in chars:
        obama_words.remove(w)
obamawords_freq = nltk.FreqDist(obama_words).most_common(10)
obama_words2 = []
for w in obama_words:
    if w not in stopwords and len(w) > 1:
        if w not in chars:
            obama_words2.append(w)
obamawords2_freq = nltk.FreqDist(obama_words2).most_common(10)
```

After running this code, I got the following results presented below:

Trump Most Popular Words: [('RT', 7944), ('Trump', 6419), ('the', 5356), ('to', 4440), ('https', 4007), ('a', 3507), ('of', 2771), ('and', 2122), ('is', 2101), ('in', 1843)]

Trump Most Popular Words(w/o Stopwords): [('Trump', 6419), ('Donald', 650), ('President', 558), ('realDonaldTrump', 499), ('Roseanne', 444), ('Shulkin', 444), ('time', 367), ('F***', 348), ('Obama', 306), ('VA', 302)]

Obama Most Popular Words: [('RT', 7919), ('the', 6161), ('Obama', 5398), ('a', 3668), ('https', 3421), ('to', 3388), ('of', 2907), ('and', 2193), ('on', 2193), ("'s", 1808)]

Obama Most Popular Words(w/o Stopwords): [('Obama', 5398), ('Trump', 1675), ('census', 777), ('FBI', 759), ('DOJ', 573), ('Rice', 550), ('Susan', 519), ('investigation', 477), ('Liberals', 474), ('Netflix', 446)]

Sub Question B Explanation:

In order to get the most frequently written hash tags I created a function titled frequent_hashtags. It took the variable containing the stored tweets as an input (ex. storedtweets_Obama). The function created an empty local variable to store the data from hashtags which was collected by using a for loop to append the text from the hashtags to this local variable. Once this was done the Counter function was used to find the most common hashtags used and the value from this was stored in another local variable, freq_hashtags which was the return value of the function.

```
91 def frequent_hashtags(tweet_list):
92     h_tags2 = []
93     for tweet in tweet_list:
94         h_tags = tweet['entities']['hashtags']
95         for tag in h_tags:
96                 h_tags2.append(tag['text'])
97     c = Counter(h_tags2)
98     freq_hashtags = c.most_common(10)
99     return freq_hashtags
```

The results received from this function for each keyword can be seen below.

Trump Most Popular Hashtags: [('Trump', 160), ('MAGA', 82), ('TheResistance', 52), ('RosanneBarr', 30), ('TrumpRussia', 29), ('FLOTUS', 25), ('BREAKING', 24), ('Mueller', 24), ('QAnon', 21), ('KeepingThemHonest', 20)]

Obama Most Popular Hashtags: [('FakeNews', 385), ('Obama', 232), ('FBR', 131), ('TheResistance', 129), ('Resistance', 129), ('Netflix', 127), ('NeverTrump', 127), ('ObamaLegacy', 127), ('ThankYouObama', 127), ('Trump', 86)]

Sub Question C Explanation:

The method I used to get the most frequently appearing usernames was a function called frequent mentions which was inspired by some code provided by professor Lee in class. It works very similarly to most of the other functions in this section of the project. It also took the variable containing the stored tweets as an input (ex. storedtweets_Obama). It then also creates an empty list to store the screen names of user mentions using a nested for loop. Finally, it implements the Counter function to determine the most commonly appearing screen names and this is returned by the function in the form of a local variable.

```
81 def frequent_mentions(tweet_list):
82     mention_list = []
83     for tweet in tweet_list:
84         mentions = tweet['entities']['user_mentions']
85         for ment in mentions:
86             mention_list.append(ment['screen_name'])
87     c = Counter(mention_list)
88     freq_ments = c.most_common(10)
89     return freq_ments
```

After calling this function on the two variables containing tweets for each keyword, I learned the top 10 most frequently appearing names for each keyword are as listed below:

Trump Most Frequently Appearing Usernames: [('realDonaldTrump', 524), ('EdKrassen', 203), ('krassenstein', 162), ('CREWcrew', 137), ('ProudResister', 132), ('thehill', 128), ('kylegriffin1', 125), ('funder', 114), ('chrislhayes', 113), ('tribelaw', 113)]

Obama Most Frequently Appearing Usernames: [('realDonaldTrump', 486), ('netflix', 423), ('mitchellvii', 421), ('RepMarkMeadows', 349), ('TheBeatWithAri', 290), ('JudicialWatch', 280), ('dbongino', 191), ('TomFitton', 190), ('StacyLStiles', 171), ('IngrahamAngle', 164)]

Sub Question D Explanation:

To find the most frequently appearing usernames, I created a function titled frequent_tweeters that worked very similarly to the frequent_hashtags function. It also took the variable containing the stored tweets as an input (ex. storedtweets_Obama). It then follows similar steps to append a local variable that contains an empty list with the screen names of twitter users. Finally, it uses the Counter function on the empty variable created and stores that in a local variable titled freq_tweeters which is the return value of the whole function.

```
101 def frequent_tweeters(tweet_list):
102     user_list = []
103     for tweet in tweet_list:
104         tweeters = tweet['user']
105         user_list.append(tweeters['screen_name'])
106     c = Counter(user_list)
107     freq_tweeters = c.most_common(1)
108     return(freq_tweeters)
109
```

After calling this function on the two variables containing the tweets for each keyword I learned that the most frequently tweeting user about Trump was an individual with the username '**ShinzonComdr**'(23 tweets) and the most frequently tweeting user about Obama was an individual with the username '**rorobin1**' (17 tweets)

Sub Question E Explanation:

To determine the influence score, I created a function called influence_score. This function took the lists of tweets defined earlier in the code process. It then went over each tweet in the list and collected text while at the same time storing influence scores. Once the list is done it returns a list with all the influence scores that were calculated in each iteration. From that you can determine what the highest influence score is and then appropriately index it to find the right tweet text.

```
111 def influence_score(tweet_list):
112     influence = []
113     tweet_txt = []
114     for tweet in tweet_list:
115         tweet_txt.append(tweet['text'])
116         quote = tweet['quote_count']
117         reply = tweet['reply_count']
118         retweet = tweet['retweet_count']
119         influence.append(quote + reply + retweet)
120     print(influence)
121     influence.sort(reverse=True)
122     return influence[0:9]
123
```

After conducting this code, I discovered that every single tweet that I collected for both of my keywords had an influence score of 0 and therefore every single tweet was equally influential.

WordCloud Step Explanation:

To create the word cloud, I first created a variable that contained the WordCloud function called with a font size of 40. I then used the generator method to generate the word cloud for each variable created earlier containing the list of words without stopwords (obama_words2 & trump_words2). I then used matplotlib to show the word clouds I created and used the function .axis to remove the axes on the Word Clouds.

```
150 #Part C. Wordcloud [20 Marks]:
151
152 wordcloud_obama = WordCloud(max_font_size=40).generate(obama_words2)
153 plt.figure()
154 plt.imshow(wordcloud_obama)
155 plt.axis("off")
156 plt.show()
157
158 wordcloud_trump = WordCloud(max_font_size=40).generate(trump_words2)
159 plt.figure()
160 plt.imshow(wordcloud_trump)
161 plt.axis("off")
162 plt.show()
```

A key note is that I was unable to install WordCloud on my computer and therefore I was instructed by Professor Lee to generate the Word Clouds below with an online generator for the purpose of this report:

Trump Word Cloud

Obama Word Cloud

Sentiment Analysis Step Explanation:

The first step for the sentiment was to define a function named collect_text which took an input of the original json file of stored tweet data, extracted raw tweet data from the original data and stored it in a new json file. The code showing how this was done can be seen presented below:

```
def collect_text(filename):
    with open(filename, 'r') as infile:
        stored_tweets = json.load(infile)
    tweet_text = []
    for item in stored_tweets:
        tweet_text.append(item['text'])
    real_filename = filename
    with open('{}_tweet_text.json'.format(real_filename[13:-5]),'w') as f:
        json.dump(tweet_text, f, indent=4)
```

The next step was to open and read the file generated in the function collect_text and store it in a variable (ex. readtweets_obama). I then created two empty lists that would be utilized to store the polarity and subjectivity scores of each tweet for each keyword (ex. pol_obama, sub_obama). Once this was done I implemented was to go over the text from every tweet for each tweet using a for loop. Within this for loop I created a variable that contained the Textblob function with the tweet text as an input (ex. tb_o = TextBlob(tweet)). Also in the same for loop, I appended each of the empty lists I had created earlier with the polarity and subjectivity scores for each tweet. The syntax for the function can be seen in the screen clipping presented below.

```
#Part D. Sentiment Analysis [20 Marks]:
collect_text('tweet_stream_Obama_10000.json')
with open('Obama_10000_tweet_text.json', 'r') as infile:
        readtweets_obama = json.load(infile)
pol_obama = []
sub_obama = []
for tweet in readtweets_obama:
        tb_o = TextBlob(tweet)
        pol_obama.append(tb_o.polarity)
        sub_obama.append(tb_o.subjectivity)
```

Once I collected all the polarity and subjectivity scores for tweets about each keyword, I created two functions make_pol_graph and make_sub_graph that make graphs of each of the results for the polarity score using matplotlib. Both functions took in the list and keyword and returned a saved pdf with a graph with the score on the x axis and the tweet count on the y axis. Each function implemented the matplotlib methods hist, xlabel, ylabel, grid and savefig. The syntax for the functions can be seen presented below:

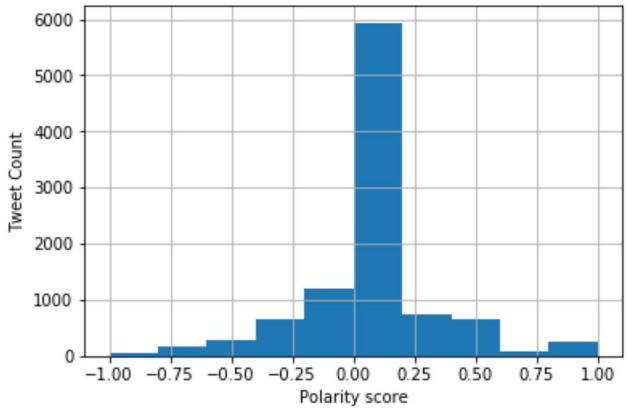
```
def make_pol_graph(lst,keyword):
    plt.hist(lst, bins=15)
    plt.xlabel('Polarity score')
    plt.ylabel('Tweet Count')
    plt.grid(True)
    plt.savefig('Polarity_{}.pdf'.format(keyword))

def make_sub_graph(lst,keyword):
    plt.hist(lst, bins=15)
    plt.xlabel('Subjectivity score')
    plt.ylabel('Tweet Count')
    plt.grid(True)
    plt.savefig('Subjectivity_{}.pdf'.format(keyword))
```

The final element was to calculate the average polarity and subjectivity scores for each keyword. To do this I created a formula that takes in the list as an input, calculates the sum of the list divided by the sum length of the list and returns this value as a result. The syntax for this can be seen below:

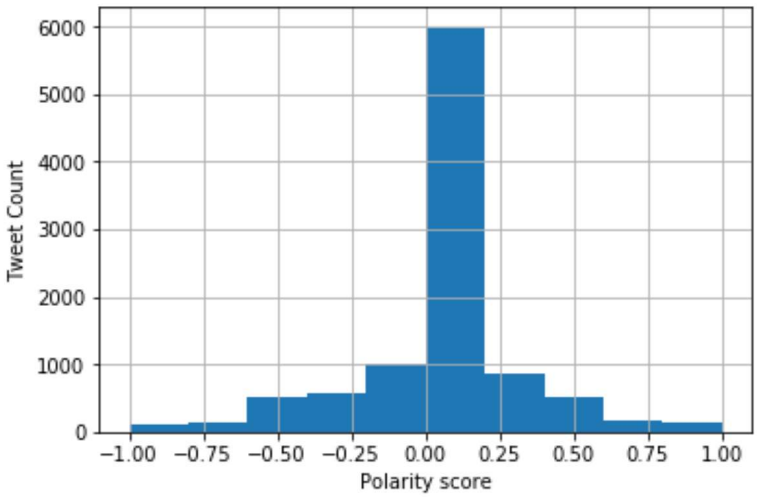
```
def compute_avg(lst):
    avg = sum(lst)/len(lst)
    return avg
```

Based on this final function called the average polarity score for trump related tweets was **0.03** and the average polarity score for obama related tweets was **0.04**. It also showed that the average subjectivity score for trump related tweets was **0.32** the average subjectivity score for Obama related tweets was **0.31**. You can also see the 4 graphs created from the code pasted below:

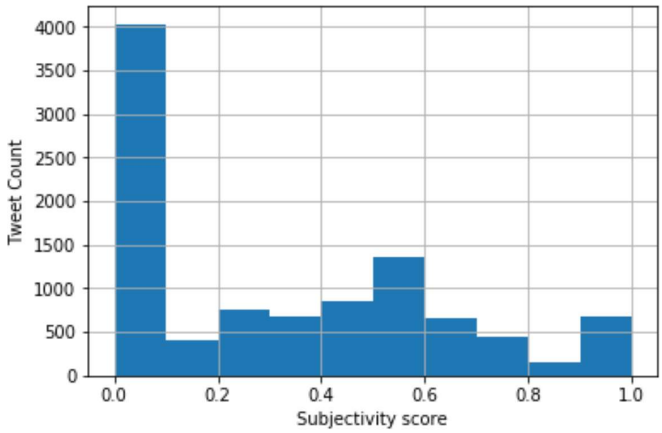
Polarity – Obama:



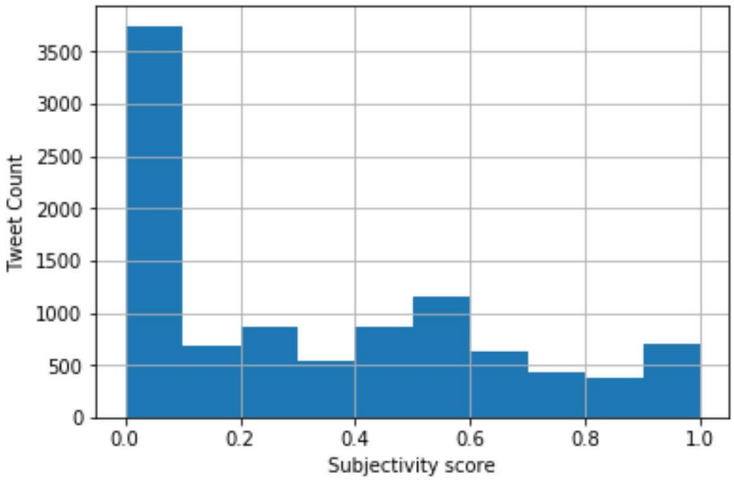
Polarity – Trump:



Subjectivity – Obama:



Subjectivity – Trump:



Insights Description:

After doing this project, I was able to learn a lot about the sentiments and discussions surrounding the keywords ‘Obama’ and ‘Trump’ . The keyword Obama had a polarity score that was slightly more positive than the keyword of Trump. It was no surprise to me that obama had a slightly more relation towards him but I was actually quite surprised that president Trump actually had a positive polarity score considering all the negative discussion surrounding his actions in the white house since he became president. It was interesting however that one of the top 10 words used in relation to trump was “F***” which represents a negative term and could represent some negative emotions towards the keyword “Trump”

One of the top 10 hashtags related to the keyword Obama were “ThankYouObama” which could suggest a level of gratitude and positivity. Another one of the top 10 hashtags was “ObamaLegacy” which could indicate that people are constantly reflecting on the what was accomplished during his term(both good and bad) and what he left behind after his term. This makes sense since he has only been out of office for a little more than a year. It was also interesting that Trump was one of the top 10 hashtags related to Obama. I believe this could mean that people make a habit of comparing the two or relating them to one another which makes sense since Trump is a successor of president obama.

One of the top 10 hashtags related to the keyword Trump was “BREAKING” which is often related news that has just come in or has made headlines. This has become one of the key characterizations of President Trump’s administration, It was also worth noting that the top mentioned username in relation to the keyword trump was “realDonaldTrump” which is his twitter handle. This may signify that people are always trying to talk to him directly or include him in their tweets specifically when they use him as the subject.