Stimulating Engagement Through Negative Emotional Sentiment on Twitter

Leslie Tucker Massad

March 20, 2018

Abstract

While it has become increasingly apparent that criticizing Donald Trump's choice of words when posting on Twitter has become a part of modern American culture, interest is raised regarding the effects and influences his purposefully negative posts have on his engagement with other public users. Through his personal account (@realDonaldTrump), our current president is using Twitter as a medium of voicing his unfiltered opinion in posts of 280-or-less characters. Unlike most celebrity figures who maintain active Twitter accounts, Trump sends tweets that are often constructed using insulting, argumentative, or otherwise negative language. Using emotional sentiment analysis to determine the overall emotional sentiment of each publication in a large set of Donald Trump's posts on Twitter, this research will primarily investigate and analyze Trump's pattern of using negative emotional sentiment on Twitter and show the advantageous outcomes that tweeting with negative emotional sentiment contributes towards his overall user engagement on the platform. In addition, this research will briefly investigate the emotional sentiment habits of other popular users on Twitter to determine if how others' public engagement is affected when posting with negative emotional sentiment.

Contents

List of Figures

List of Tables

1 Introduction

In a time where social media exists as a primary source of communication, Twitter is used by president Donald J. Trump predominantly to publicly describe his attitude and viewpoints towards modern political affairs. As the president 45th president of the United States, Trump's Twitter account is put under a distinct spotlight as his posts will be read by several millions of users. Despite this position of high attention, Trump continues to post tweets that contain moderate to severe negative emotional sentiment in his regular posting regimen of $\sim$14 posts per day. Raising the question of why Trump would uphold a habit that would intuitively suggest a harmful impact on his likability, this research is aimed to examine and confirm the contrary - tweeting with negative emotional sentiment in fact increases how often and how many users engage with his posts. Revealing this insight gives us a better understanding of the incentives behind regularly posting with negative emotional sentiment as well as the unexpectedly fruitful impact it has on one's overall user engagement on Twitter.

2 Background and Related Work

There are several realms of research that directed my interest towards Big Social Data extraction and analysis alongside Natural Language Processing of Twitter data. My initial research began with looking at Big Social Data in a broad sense, meaning the accessibility, functional usability, and applicability of social data at a large scale. The paper Ekaterina's paper *Conceptualizing Big Social Data* [11] gives an ample description of the usefulness of social media data at a large scale from a plethora of platforms (Facebook, Twitter, Instagram, etc.). There are several different unique components to each platform that influence how the information provided can be extracted and analyzed. Pictures, links to additional web pages, and other media can all be relevant to certain niches of study but my interest gravitates towards the analyzation of the english language. Additionally and more importantly, the patterns that can be observed when language data is gathered at a large scale can be incredibly useful to determine a majority attitude about a certain topic. As well, Suthakar's study *An efficient strategy for the collection and storage of large volumes of data for computation* [15] provided plentiful background information on strategies and proper methods to create Big Social Data sets that can efficiently be analyzed. This paper also brought forth the scale of usefulness when it comes to analyzing very large amounts of data - leading my attraction towards language information because of its ease of analyzation. Knowing Twitter is predominantly language data, my interest grew in using Twitter posts as data to answer valuable questions about trends in our society.

Knowing I wanted to use Twitter as my platform of information extraction, my research shifted towards

studies that have efficiently created data sets from Twitter posts. Padmaja's paper *TreeNet Analysis of Human Stress Behavor using Socio-Mobile Data* [12] brought forth useful information in terms of what it is like to not only obtain unique Twitter posts that hold specific value at a large scale, but also it explains the process of analyzing language to determine human emotion within posts. While this study spends a large amount of effort in describing the usability of machine learning algorithms to analyze human expression over tweets, simply put the research looked to conclude if a user expressed stress when posting to Twitter. My interest was similar in the fact that I wanted to use language in Twitter posts to determine what kind of emotional expression is contained within tweets at a large scale, but I sought to dive deeper into this area of study. Priya's paper *Social Media, Big Data, and Public Health Informatics: Ruminating Behavior of Depression Revealed through Twitter* [10] solidified my interest in extracting useful information from the types of language used within posts on Twitter. Looking to find tweets with keywords that are often related to signs of depression, this study embarked on similar efforts to count how often a user posted using these keywords. Rather than simply determining if the language in a post shows a certain category of emotional expression, my original thoughts were to set out to bring this big data analyzation a step further by determining the level of intensity of emotional expression. More specifically my plan of research would build upon these past studies of extracting emotional expression by quantifying different intensities of emotional sentiment. Yet what became more interesting as my research continued is what effects and influences using negative emotional sentiment versus using positive sentiment in a user's Twitter posts have on how others engaged with the user on the platform. More specifically I have tasked myself with answering the question of whether using negative emotional sentiment, something that seemingly would promote poor reactions from other users, in fact has positive effects on one's user engagement such as promoting above-average support.

To continue, Mahmood's paper *Mining Twitter big data to predict 2013 Pakistan election winner* [9] brought my research interest towards the realm of politics and the applicaiton of Twitter analysis on popular government figures. This study similarly looked to determine emotional sentiment from Twitter posters of Pakistan prior to the election of their president in 2013. By quantifying emotional expression in posts that mention a specific presidential candidate and the measuring shifts in emotional expression over time, researchers were able to accurately make assumptions about the popularity of a candidate and the expected outcome of an election.

In continuation, Kaguera and Sunghwhan's paper *Twitter Content Eliciting User Engagement: A Case Study on Australian Organisations* [8] solidified my interest in using engagement as an output variable for my study. Using the number of favorites, retweets, and replies as x variables to determine the level of engagement with a tweet, I tasked myself with using similar variables for measurement as well as furthering the extent in which engagement can be measured by using additional variables such as the number of fol-

lowers gained and the number public mentions. Another paper that used similar variables in determining user engagement on Twitter is Hu, Farnham, & Talamadupula's paper *Predicting User Engagement on Twitter with Real-World Events* [5] in which the total number of retweets, direct communications, and broadcast communications were recorded alongside the maximum tweets per hour from a user. While these variables aren't directly relatable to the study I will be conducting, it was very helpful in obtaining wholistic knowledge about engagement and honing in on the variables that are most applicable for my study. Lastly, Junco & Loken's paper *The effect of Twitter on college student engagement and grades* [7] was helpful to read as well as it added to the relevance of the insights that can be drawn from Twitter usage.

3 Methods and Design

3.1 Obtaining the Data

This research is decisively split into two unique data sets in which seperate methods of extraction are required for each of the two sets. The first data set includes historical tweets sent exclusively by Donald Trump in the past several years that amounts to about 17,000 total tweet documents. It is important to note that this initial data set will be collected using the normal Twitter API in which tweet attributes such as the number of 'favorites' and the number of 'retweets' that a tweet recieved holds relevence due to the fact that the target tweets were posted in the past. The quantity of favorites and retweets a tweet has make up two of the five quantifications we will use to judge how much engagement Donald Trump has drawn to the public Twitter audience with a specific tweet. The second data set will be collected using Twitter's Streaming API in which not only Donald Trump's tweets are collected in real-time, but all public replies to Trump's tweets are collected as well alongside all public tweets that include a mention of Trump by name (ex. 'trump', 'Donald Trump'). Using the streaming API will allow us to make an additional three quantifications that we will later analyze in order to better understand how engaged Twitter users are with Donald Trump's tweets: the frequency of replies sent to individual tweets posted by Donald Trump, frequency of public tweets mentioning Trump by name after Donald Trump posts a tweet, as well as the rate in which Trump gains additional followers after a specific tweet. Using these five total quantifications, we will compare the differences in these values as Trump tweets both negatively and positively.

In order to obtain the first data set we first use the web scraping tool Selenium 2.48.0 [6], a python-integrated web scraping tool that imitates a user browsing on a specific internet explorer, to obtain the unique identifier string for each tweet we'd like to obtain. Using these identifiers, we'll extract all information provided by the Twitter API for each tweet (text, date created, source, etc.) and save this information

to a database as individual JSON objects. I created a python script that saves each unique identifier string given to tweets posted by @realDonaldTrump within a certain time range [view Appendix A]. This script functions by continuously loading single webpages of all tweets sent by @realDonaldTrump given an exact date, adding each tweet's identifier from that day to a list. Once each page containing all tweets from a specific day between a given range (June 16, 2014 - Oct 01, 2017 in our case) has been saved, we'll have successfully obtained each tweet identifier string for each of our desired tweets for our first data set. These identifier strings are separated by "," and saved to a text file that will be used for extracting each tweet object. It is important that Selenium is used to obtain the list of all tweet identifiers in this study as opposed to directly grabbing tweet objects via Tweepy because our data set will contain $>$ 3,200 tweet objects - the Twitter API's limit on successive get calls from a single user's timeline. Once this desired list is obtained we then use Tweepy 3.5.0 [13], a python wrapper for accessing the Twitter API, to make successive get calls to obtain each tweet object given an associated identifier in our list of identifiers. Each call returns a JSON object representing all information given by the Twitter API associated with a specific tweet. Since we already know the identifier for each tweet we'd like to obtain, we can utilize the *wait_on_rate_limit* parameter when initializing the Tweepy extractor to extract each tweet successively while also avoiding the program from terminating upon reaching the Twitter API's get time window limit - 180 calls every 15 minutes. In addition, these tweets are stored in a database using CouchDB 2.1.0 in order to maintain an easily accessible and organized list of our tweet objects. Each JSON object is extracted and saved into the CouchDB database within the same function, completing the extraction of each desired tweet for our first data set. [view Appendix B].

Obtaining the second data set does not require the use of Selenium, but it is important to note the breadth of space needed ($>$ 10 GB) to hold this data set as it will contain several millions of tweet documents. Additionally, this extraction method to obtain the second data set requires the use of a virtual machine in order to gaurentee uninterrupted execution of a python script that is purposefully set to run for an indefinite amount of time. This second script [view Appendix G] utilizes Streaming Twitter API functions in which all target tweets are extracted in real-time as they are posted without any limititations on the quantity of tweets we extract. Replies are collected and saved to a seperate database than the collected public mentions (*trumpstream* and *talkstream* respectively), yet this seperation is not neccassary if the collection process were to be imitated. Another important note to make about the streaming collection process is the use of several exceptions (*ProtocolError, IOError, AttributeError, etc.*) within the infinite *while(True)* loop that wraps all essential extraction code within. Since we are interested in keeping the code from stalling if it runs into an unparsable tweet, it is important to catch all possible exceptions in order to log them and more importantly

continue extraction thereafter. In my study this script was run for just over 14 days and amounted to ∼6.5 million tweets.

3.2 Organization of Engagement Attributes

Now that we have each desired tweet saved onto the CouchDB databases, our focus turns towards analyzing the historical tweets within the first database. For this analysis we will look at a select few attributes of each tweet object:

- "created_at" - $string$ object containing the date, time a tweet was posted
- "text" - $string$ object containing all characters written in a tweet
- "favorite_count" - int object containing the total amount of favorites a tweet has received
- "retweet_count" - int object containing the total amount of retweets a tweet has received

The first mentioned attribute $created\_at$ is used to organize the data set into three distinct subsets by the date each tweet was posted. Because the string is formatted in a way that gives more details than we need (ex."Thu Oct 16 17:18:40 +0000 2014"), a helper function to reformat the date into a shorter month/day/year (ex. "Dec 23 2014") format is used [view Appendix C]. Since we can easily produce a neatly formatted date for the time of creation on each tweet, we'll now begin to organize our tweets into the following three subsets in relation to Trump's political candidacy of the time:

(i) *Pre-Campaign* (June 16, 2014 - June 16, 2015)

(ii) *Campaign* (June 17, 2015 - Nov 07, 2016)

(iii) *President* (Nov 08, 2016 - Oct 01, 2017)

Two boolean helper functions are necessary in order to confirm if a tweet was made during the *President* or *Campaign* period. If a tweet's date is not confirmed to be in either of the two period's, we know that it belongs in the *Pre-Campaign* period [view Appendix D]. The data set is split into these three periods in order to better observe Trump's growth of overall engagement over the course of his political campaign. It is to be assumed that his Twitter account has grown a significant amount in terms of user engagement over the course of his political career and we will later observe this to be true. These helper functions allow us to create three iterable lists of tweets that share a creation date within the same time period. While these groupings are important for understanding Trump's account growth over time, the data will be analyzed holistically as well to determine more general patterns of the levels of user engagement he receives.

Next we'll utilize the *text* attribute in order to determine an emotional sentiment score for each tweet. Using the MPQA Subjectivity Lexicon [17] for reference, a function that scans each individual word in a tweet's text and checks if there exists a correlated sentiment (positive or negative) with the word is used to score a tweet's overall emotional sentiment [view Appendix E]. Each word that has confirmed negative or positive connotation attributes -1 or +1 sentiment score respectively. We'll refer to *negative* tweets as posts that contain text with an overall sentiment score < 0 and *positive* tweets as posts that contain text with a score > 0. This scoring function will allow us to calculate average emotional sentiment for specific tweets within a certain time period as well as throughout the whole data set. Using figures 1,2 we can observe both a negative tweet and positive tweet being scored using the lexicon:

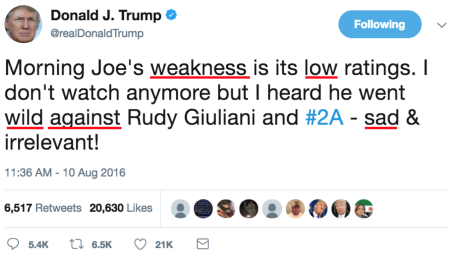


Figure 1: Ex. Negative Tweet [Score = -5]

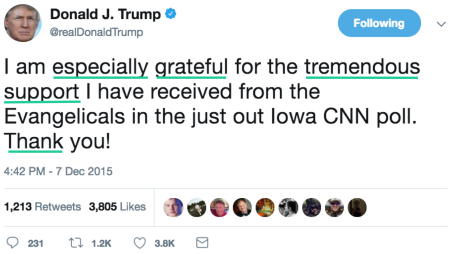


Figure 2: Ex. Negative Tweet [Score = +5]

In continuation *favorite_count* and *retweet_count* will be used to calculate the average number of favorites and retweets Donald Trump's posts receive within each time period as well as throughout the entire first data set. To obtain these averaged counts for each time period, two simple functions are created where a running sum of favorites or retweets received on each tweet within a period is divided by the total number of days the respective period represents [view Appendix F]. These quantifications represent the first two of five calculations we will make to judge the level of engagement Trump's tweets recieve.

Moving to Streaming Data, the remaining three quantifications we will make will be refered to as the following: *reply count*, *followers gained*, and *public mention count*. In order to obtain the quantity of replies that a tweet sent from Donald Trump recieves, we must utilize the *in_reply_to_status_id* JSON attribute that each tweet holds. It is important to mention that our *reply count* quantity will represent only the amount of replies one of Trump's tweets recieves in the 24-hour period after the tweet has been sent. This time limitation is included in order to allow for an equal representation of the amount of replies each tweet recieves. If a public tweet that we have collected is in fact a reply to a specific tweet made by Trump, this field will contain the tweet *id_str* number of the unique tweet Trump posted (and *NULL* if it is not a reply). Using this information, we can simply count each of the tweets in our second data set that has its

in_reply_to_status_id set the desired tweet sent by Trump [see appendix H].

Next, obtaining the amount of followers Trump gains after sending a tweet for our *followers gained* quantification requires another function that simply takes the difference between the *follower_count* JSON value in an initial tweet and the *follower_count* of a tweet sent 24-hours after the initial tweet. Using this information, we will be able to calculate how many followers-per-hour Trump gains between the period of two tweets by dividing the difference in the *follower_count*'s by the amount of time spent between sending the two tweets. Once this value is calculated, we take the followers-per-hour rate and multiply it by 24 in order to accurately estimate how many followers Trump has gained in the 24-hour period after sending a tweet [see Appendix I]. Again, a 24-hour time limitation is used when quantifying our *followers gained* estimation because we want to equally judge the amount of followers gained after each tweet. An important note to make about this function is that the *followingTweet* paramenter must be specifically chosen to be a tweet made at least 24-hours after the original tweet in order to make sure the calculation is accurate.

Lastly, in order to obtain the amount of public mentions that are posted after Trump has sent a tweet to calculate our *public mention count* variable, a final function is created to simply count each individual public tweet mentioning Trump by name that has been made 24-hours after a Trump has sent a tweet. Since we have placed each possible public mention into a seperate database, this function simply counts each of the individual tweets in the database that was sent in the 24-hour period after Trump tweeted [view Appendix J]. Using this information, we can accurately determine the total amount of public mentions made the following day after Trump has tweeted.

4 Data Sets and Analysis

4.1 Data Set 1 - Historical Tweet Data

Total Number of Tweets: 16,943

- Pre-Campaign (June 16, 2014 - June 16, 2015)
    - Average Tweets per Day: 21.3
    - Average 'favorites' per Tweet: 4,702
    - Average 'retweet' per Tweet: 1,583
- Campaign (June 17, 2015 - Nov 07, 2016)
    - Average Tweets per Day: 13.0

    * Average 'favorites' per Tweet: 8,706
    * Average 'retweet' per Tweet: 3,105
- President (Nov 08, 2016 - Oct 01, 2017)
    * Average Tweets per Day: 6.8
    * Average 'favorites' per Tweet: 85,603
    * Average 'retweet' per Tweet: 19,709

4.2 Data Set 2 - Streaming Tweet Data

Total Number of Tweets: 6,611,232

- Total Replies Collected: 2,845,039
    - Average Replies made 24-hours after a tweet : 13,911
- Total Public Mentions Collected: 3,766,193
    - Average Public Mentions 24-hours after a tweet: 232,605
- Average Followers Gained 24-hours after a tweet: 43,275

4.3 Measuring Engagement: Negative vs Positive

Using the functions I have created, we will start our analysis by calculating the differences in the average number of favorites and retweets received on both negative and positive tweets. It is important to understand what both a favorite and retweet entails when a user willingly marks a tweet with either of the two options. A favorite, analogous to a 'like' on other social media platforms such as Facebook, adds the tweet to a user's publicly viewable list of supported tweets. While a user's decision to favorite a tweet doesn't necessarily mean they agree/disagree with the originator of the tweet, it is an accurate way to determine whether a user has read the tweet on top of letting other user's know that the user at hand consents to having the tweet added to their personal list of favorited tweets. Through this fact we are able to generalize that users who decide to favorite a tweet confirm that they have made a sufficient level of engagement with the post and thus the user can be assumed to inherently feel stronger likability towards the tweet than other posts they have not decided to favorite.

In addition, a retweet represents a user consenting to display a certain tweet on their personal timeline. Unlike a favorite, a user's decision to retweet a tweet is a strong display of support for the tweet as the

post will be displayed not only on their user timeline but all users following the user who has retweeted the tweet will see the post on their personal Twitter feed. With this in mind, we can safely make the assumption that a user's decision to retweet a post entails that they have read and maintained a sufficient level of engagement with the tweet.

Using the quantity of favorites and retweets a tweet maintains as a measurement of the level of public engagement the tweet received, we'll look at the averages for each of the three time periods. It is important to note that tweets which have an emotional sentiment score $\neq 0$ are considered in the following statistics in sections 4.2.1, 4.2.2.

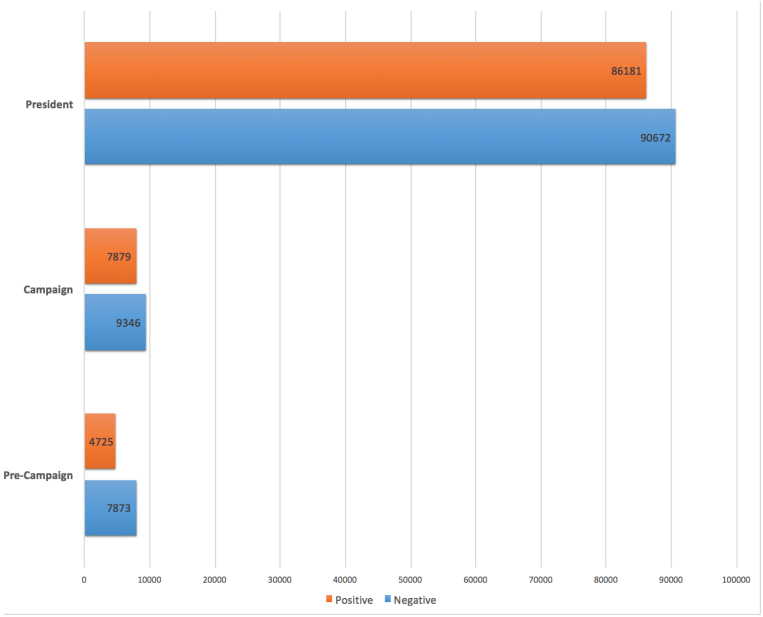
4.3.1 Average Favorites

Average quantity of favorites per tweet within each of the three time periods can be observed as follows:

- Pre-Campaign
    - Negative: 7,873
    - Positive: 4,725
- Campaign
    - Negative: 9,346
    - Positive: 7,879
- President
    - Negative: 90,672
    - Positive: 86,181

We can observe that throughout all three periods, Donald Trump's average number of favorite's per tweet is 3,045 higher when the tweet carries negative emotional sentiment compared to tweets with positive sentiment (Pre-Campaign [+3,148], Campaign [+1,497], President [+4,491]). Figure 3 portrays a visualization of this data.

Figure 3: Average Likes - Negative vs Positive



4.3.2 Average Retweets

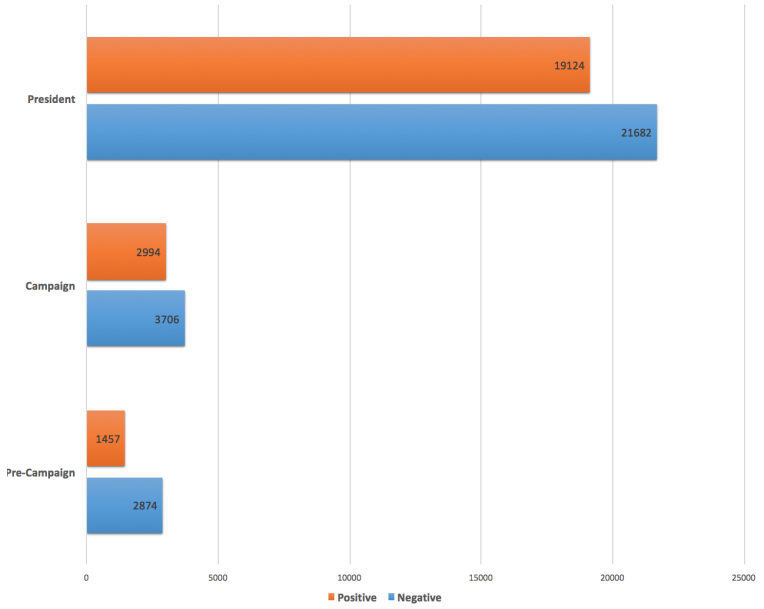
Average quantity of retweets per tweet within each of the three time periods can be observed as follows:

- Pre-Campaign
  - Negative: 2,874
  - Positive: 1,457
- Campaign
  - Negative: 3,706
  - Positive: 2,994
- President
  - Negative: 21,628
  - Positive: 19,124

Alike to favorites, we can observe that the average quantity of retweets received on Donald Trump's posts containing negative emotional sentiment is higher than those containing positive sentiment. Throughout all time periods, the average number of retweets per tweet is 1,544 higher for posts with negative

emotional sentiment (Pre-Campaign [+1,417], Campaign [+712], President [+2,504]). Figure 4 portrays a visualization of this data.

Figure 4: Average Retweets - Negative vs Positive



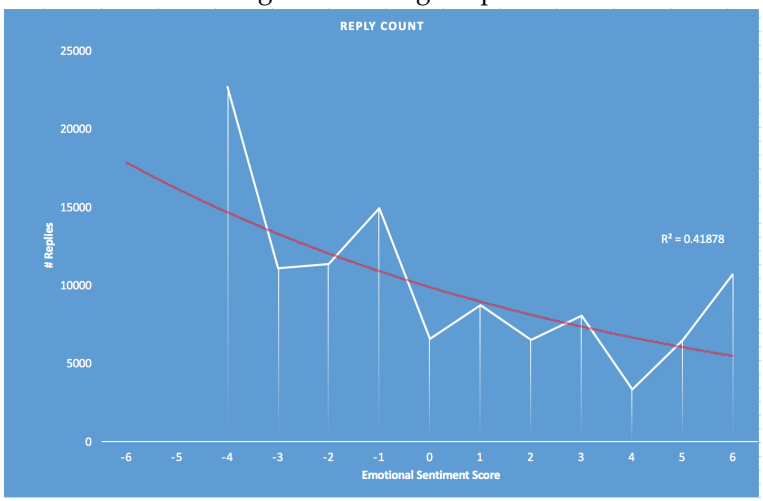
4.3.3 Average Replies

Replies sent to a tweet posted by Donald Trump represents a Twitter user's willingness to publicly voice their opinion about a tweet sent by our preseident and/or send a tweet that is directly targetted towards Trump. Since a reply requires a user to take the time to create and send a tweet directed at Trump specifically, we can conclude it is an accurate representation of a user engaging with one of Donald Trump's tweets. After calculating the quantities of replies recieved in the 24-hour period after Trump has tweeted, we can observe the average quantities of replies accumulated for tweets sent by Trump with both negative and positive sentiment as follows:

- Average Reply Count 24-hours after tweeting with **negative** sentiment: 20,002
- Average Reply Count 24-hours after tweeting with **positive** sentiment: 7,821

Not only can we observe that Trump's tweets with negative sentiment recieve a substantially larger amount of replies in the 24-hour period after posting, but we can observe a linear relationship between the extent of negativity used and the amount of replies recieved. Looking at Figure 5, the x axis represents the emotional sentiment score of a tweet sent by Trump and the y axis represents the amount of average replies recieved 24-hours after tweeting with a certian sentiment score:

Figure 5: Average Replies



Additonally after running a two-tailed t-test where our null hypothesis is equal to [$\mathcal{H}$ = Reply counts are the same for negative and positive sentiment tweets], we can affirmitevly reject the null hypothesis as the test provides us with a p-value of 0.028. With this information we are able to definitely conclude that the amount of replies Trump recieves on a tweet of negative emotional sentiment is statistically different than the amount of replies he recieves on tweets of positive emotional sentiment.

4.3.4 Average Followers Gained

A user's decision to follow Donald Trump's Twitter account is another measurement of engagement because they are willingly deciding to have all future tweets sent by Trump be included in their personal timeline. For each day that the streaming collection script was running and collecting *follower count* information, two groupings were created in which we categorize the overall emotional sentiment Trump's tweets represent for a given day:

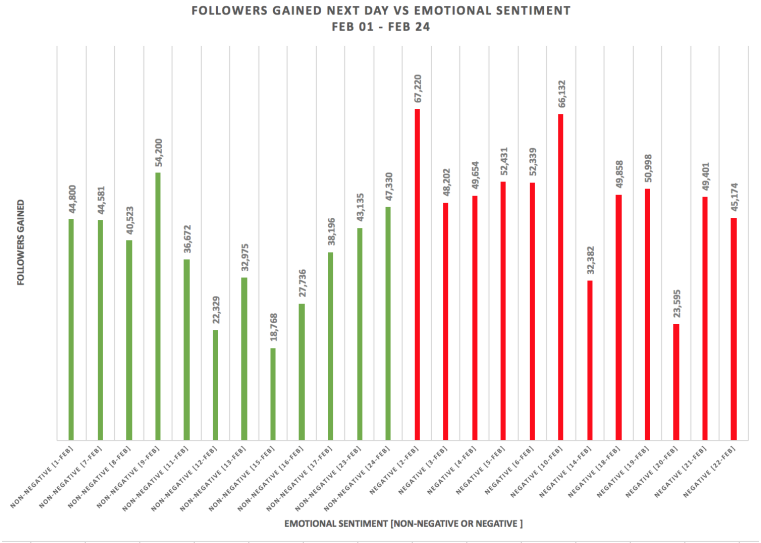
- Negative - A day in which Trump has tweeted with negative emotional sentiment at least once (Sentiment Score < 0).
- Non-Negative - A day in which Trump has tweeted with exclusively neutral or positive emotional sentiment (Sentiment Score ≥ 0).

The amount of followers gained 24-hours after each tweet sent during a day of both negative and non-negative groupings is averaged, giving us the average quantitiy of followers gained during a specific day. With this information we are able to calculate the average amount of followers gained after both negative and non-negative days as follows:

- Average followers gained 24-hours after tweeting **negatively**: 48,948
- Average followers gained 24-hours after tweeting **non-negatively**: 37,603

Visualized in Figure 6, there were a total of twenty four full days (Feb 1 - Feb 24) in which *follower count* information was extracted. There were a total of twelve days in which Trump tweeted negatively at least once, as well as a total of twelve days in which Trump tweeted exlclusively non-negatively.

Figure 6: Average Followers Gained



Using this information we are able to determine that when Trump tweets with negative emotional sentiment at least once during a given day, he will recieve an average of 11,345 additional followers than a day in which he tweeted exclusively non-negatively.

4.3.5 Average Public Mentions

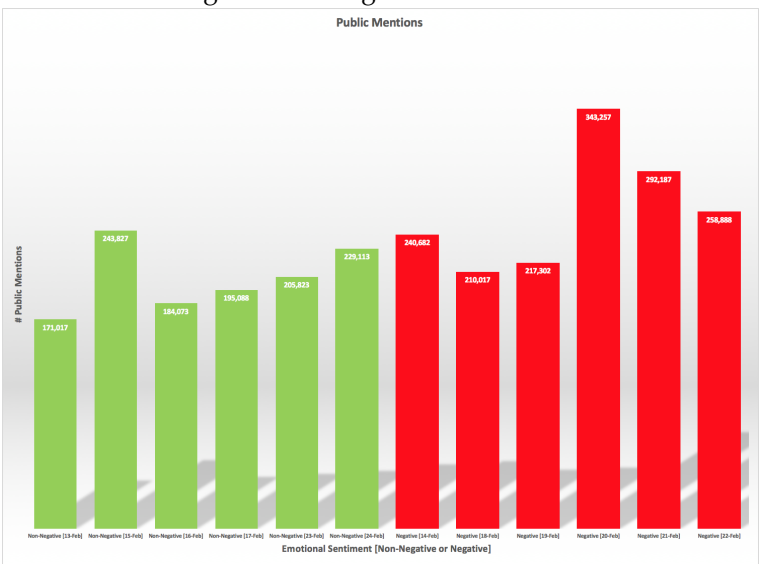
Our last quantificaiton we will analyze in relation to emotional sentiment of Trump's tweets is the *public mention count* attribute. The amount of public tweets sent on Twitter than include a mention of Donald Trump's name during a given day is useful to our judgement of engagement due to the fact that a user's decision to mention our president's name in their tweet means they are directly or indirectly talking about their opinion about Trump. Using the same categorization technique used in *follower count* analysis determining the overall sentiment of Trump's tweets given a specific day (negative or non-negative), below describes the amount of average replies recieved for each of the days *public mention count* information was being extracted:

- Average public mentions during a day Trump tweeted **negatively**: 260,388

- Average public mentions during a day Trump tweeted **non-negatively**: 204,823

Visualized in Figure 7, public mentions were extraced for a total of twelve days (Feb 13 - Feb 24). Of those twelve days, there were six days in which Trump tweeted with negative emotional sentiment at least once and six days in which Trump tweeted exclusively non-negatively.

Figure 7: Average Public Mentions



With the following information we can determine that on days in which Trump tweeted negatively at least once, he avergaed an additional 55,565 public mentions than on days in which he tweeted non-negatively.

5 Addtional Subjects

While we have a substantial amount of information on the engagement trends of Donald Trump's tweets when he posts with positive and negative emotional sentiment, running prelimanary tests on additional celebrity-status Twitter users will allow us to determine the unanimity of the trend in which negative tweets produce more engagement than their positive counterparts. Three additional subjects were chosen and the most recent 200 tweets sent by each subject were extracted. We can observe information on these three addtional subjects as follows:

- Daniel Tosh (@danieltosh)
  - Followers: 27.4 million
    - Average favorites per **negative** tweet: 4,421

        * Average retweets per **negative** tweet: 954
        * Average favorites per **positive** tweet: 2,466
        * Aveage retweets per **positive** tweet: 490
    - Average Tweet Sentiment: 31% Negative, 69% Positive
- Aubrey Graham (@Drake)
    - Followers: 36.7 million
        * Average favorites per **negative** tweet: 27,046
        * Average retweets per **negative** tweet: 30,819
        * Average favorites per **positive** tweet: 21,450
        * Aveage retweets per **positive** tweet: 18,242
    - Avergae Tweet Sentiment: 24% Negative, 76% Positive
- Lebron James (@KingJames)
    - Followers: 40.8 million
        * Average favorites per **negative** tweet: 32,388
        * Average retweets per **negative** tweet: 7,199
        * Average favorites per **positive** tweet: 28,073
        * Aveage retweets per **positive** tweet: 6,247
    - Average Tweet Sentiment: 28% Negative, 72% Positive

We can observe that for all three additional subjects, tweets sent with negative emotional sentiment received on average a larger amount of favorites and retweets than tweets sent with positive emotional sentiment. While this information does not include as many tweets as we analyzed in our first data set of Trump's historical tweets, this information is useful as we can determine our findings may be unanimous to all popular users on Twitter.

6 Conclusions

Using the information from section 4, we can conclude that a trend exists in which tweets that Donald Trump sends that include negative emotional sentiment will recieve more engagement than tweets he sends constructed using positive emotional sentiment. For all of the five quantificaitons made that have

be analyzed in this study, comparing each of the outputs when grouping negative and positive emotional sentiment remained in the same trend in which negativity brought forth more public engagement than positivity. Moreover, this study concretely affirmed that there is a statistical significance between the difference in the amount of replies Trump's tweets of negative emotional sentiment recieve compared to tweets sent with positive emotional sentiment. In addition, the information from section 5 provides us with telling evidence that this pattern of negative emotional sentiment producing additional engagement compared to positive sentiment may be unanimous to popular users on Twitter.

7 Future Work

There are several things to be done in the continuation of this project that will allow us to further analyze the impact of using both negative and positive emotional sentiment on one's public engagement on Twitter. First, analyzing sentiment in terms of absolute emotion used (meaning both negative and positive words attribute to an increase in sentiment score) would be very interesting as we would be able to answer questions such as "How *much* emotion (negative or positive) outputs the most public engagement?" or "What kinds of combinations of positive and negative words output the most public engagement?". Determining an ideal level of emotional sentiment would be an interesting extension of this study as I would hypothesize that extreme levels of negativity or positivity would hinder ones engagement with the public.

Another extension of this study that would provide us with worthwhile information would be to analyze the emotional sentiment of replies sent to Trump's tweets and the relationship between the emotional sentiment of public replies and the emotional sentiment of a tweet Trump sends. I would hypothesize that the higher level of negative emotional sentiment that a tweet from Trump includes, the higher the level of negativity in public replies - yet this is not something I tasked myself with in this study. Additionally, the same sentiment analysis can be done for public mentions as knowing how public users react emotionally in their replies/public mentions would be interesting to calclulate.

Lastly, an additional extension of this project could be to analyze less-popular users on Twitter to see if the same trend applies to users who do not have as large of a public following as the users I analyzed in this study. This study singled-out very popular users on the platform due to the fact that it allows for larger deviations in the quantifications we made, yet it would be very interesting to conduct similar analysis on users who don't have millions of followers as it may output different results.

Appendices

A Code Example 1 [Historical Tweet ID Collection]

```
#@author Tucker Massad
#@author bpb27
#https://github.com/bpb27/twitter_scraping/blob/master/scrape.py

from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import NoSuchElementException,
    StaleElementReferenceException
from time import sleep
import json
import datetime

user = 'realDonaldTrump'
start = datetime.datetime(2014, 6, 16) # year, month, day
end = datetime.datetime(2017, 10, 10) # year, month, day
delay = 1 # time to wait on each page load
driver = webdriver.Chrome() # choice of web browser

twitter_ids_filename = 'all_ids.json'
days = (end - start).days + 1
id_selector = '.time a.tweet-timestamp'
tweet_selector = 'li.js-stream-item'
user = user.lower()
ids = []

def format_day(date):
   day = '0' + str(date.day) if len(str(date.day)) == 1 else str(date.day)
   month = '0' + str(date.month) if len(str(date.month)) == 1 else str(date.month)
   year = str(date.year)
   return '-'.join([year, month, day])
```

```
def form_url(since, until):
    p1 = 'https://twitter.com/search?f=tweets&vertical=default&q=from%3A'
    p2 = user + '%20since%3A' + since + '%20until%3A' + until +
        'include%3Aretweets&src=typd'
    return p1 + p2


def increment_day(date, i):
    return date + datetime.timedelta(days=i)


for day in range(days):
    d1 = format_day(increment_day(start, 0))
    d2 = format_day(increment_day(start, 1))
    url = form_url(d1, d2)
    print(url)
    print(d1)
    driver.get(url)
    sleep(delay)

    try:
        found_tweets = driver.find_elements_by_css_selector(tweet_selector)
        increment = 10

        while len(found_tweets) >= increment:
            print('scrolling down to load more tweets')
            driver.execute_script('window.scrollTo(0, document.body.scrollHeight);')
            sleep(delay)
            found_tweets = driver.find_elements_by_css_selector(tweet_selector)
            increment += 10

        print('{} tweets found, {} total'.format(len(found_tweets), len(ids)))
```

```
        for tweet in found_tweets:
            try:
                id =
                    tweet.find_element_by_css_selector(id_selector).get_attribute('href').split('/')[-1]
                ids.append(id)
            except StaleElementReferenceException as e:
                print('lost element reference', tweet)

    except NoSuchElementException:
        print('no tweets on this day')

    start = increment_day(start, 1)

try:
    with open(twitter_ids_filename) as f:
        all_ids = ids + json.load(f)
        data_to_write = list(set(all_ids))
        print('tweets found on this scrape: ', len(ids))
        print('total tweet count: ', len(data_to_write))

except (OSError, IOError):
    with open(twitter_ids_filename, 'w') as f:
        all_ids = ids
        data_to_write = list(set(all_ids))
        print('tweets found on this scrape: ', len(ids))
        print('total tweet count: ', len(data_to_write))

with open(twitter_ids_filename, 'w') as outfile:
    json.dump(data_to_write, outfile)

print('All IDs written to file')
driver.close()
```

B Code Example 2 [Historical Tweet Collection]

```
#@author Tucker Massad
#Constructor to access Twitter API. Personal consumer key, consumer secret, access
    token, and access secret must be used. I suggest storing these variables in a
    seperate file, settings.py
def setupExtractor():

   auth = OAuthHandler(settings.consumer_key,
                  settings.consumer_secret)

   auth.set_access_token(settings.access_token,
                   settings.access_secret)


   api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)


   return api


extractor = setupExtractor()
server = couchdb.Server()
db = server[settings.database]

#Using a list of tweet IDs, save each tweet object to the database
def saveTweets():

   alltweets = []

   with open('all_ids.json') as data_file:
      all_json_ids = json.load(data_file)
      totalIdNumber = 16942
      sofar = 0
```

```
        while sofar <= totalIdNumber:
            specificTweet = extractor.get_status(id=all_json_ids[sofar])
            alltweets.append(specificTweet)
            print sofar
            sofar += 1

        totalsaved = 0
        for tweet in alltweets:
            db.save(tweet._json)
            totalsaved+=1

    return totalsaved, "tweets saved to the database!"
```

C Code Example 3 [Formatted Date]

```
#Returns the month, day, and year of a tweet in MMM/DD/YYYY format.
#@param date: A tweet objects "created_at" attribute
def getFormattedDate(date):
    month_and_day = date[4:10]
    year = date[26:]
    return month_and_day + " " + year
```

D Code Example 4 [Historical Tweet Organization]

```
#@author Tucker Massad
#Returns list of all tweets from Jun 16, 2014 - June 16, 2015
def getCampaigningTweets():
    campaigningTweets = []
    tweetCount = 0
    for id in db:
        tweet = db.get(id)
        date = tweet.get("created_at")
```

```
        formattedDate = getFormattedDate(date)
        if confirmCampaigningPeriod(formattedDate):
            campaigningTweets.append(tweet)
            tweetCount += 1
    return campaigningTweets




#Returns list of all tweets from Nov 08, 2016 - Oct 01, 2017
def getPresidentialTweets():
    presidentialTweets = []
    tweetCount = 0
    for id in db:
        tweet = db.get(id)
        date = tweet.get("created_at")
        formattedDate = getFormattedDate(date)
        if comfirmPresidentPeriod(formattedDate):
            presidentialTweets.append(tweet)
            tweetCount+=1
    return presidentialTweets




#Returns list of all tweets not in getCampaigningTweets() and getPresidentialTweets()
def getPreCampaignTweets():
    precampaignTweets = []
    tweetCount = 0
    for id in db:
        tweet = db.get(id)
        date = tweet.get("created_at")
        formattedDate = getFormattedDate(date)
```

```
      if not comfirmPresidentPeriod(formattedDate):
         if not confirmCampaigningPeriod(formattedDate):
            precampaignTweets.append(tweet)
            tweetCount+=1
   return precampaignTweets
```

E Code Example 5 [Emotional Sentiment Scoring]

```
#@author Tucker Massad
#Returns overall emotional sentiment score given a tweets text
#@param tweet: A tweet objects "text" attribute
#@param adjusted: True if you would like to score total emotional recognition
def getEmotionalScore(tweet, adjusted):
   score = 0
   splitTweet = tweet.split()
   emotionDictionary = getEmotionalDictionary()
   for word in splitTweet:
      for wordAndScore in emotionDictionary:
         for key, value in wordAndScore.items():
            if word == key:
               if value == "positive":
                  score+=1
                  #print "Plus score! Word used:", key, " Sentiment: ", value
               elif value == "negative":
                  if adjusted == True:
                     score += 1
                  else:
                     score -= 1
                  #print "Minus score! Word used:", key, " Sentiment: ", value


   return score
```

```
#Using the MPQA emotional sentiment lexicon file, return dictionary entry for a word
    and its associated sentiment score
def getEmotionalDictionary():
   f = open("emotionalLexicon.tff", "r")
   wordToScore = []
   for line in f.readlines():
      splitwords = line.split()
      current_word = splitwords[2]
      current_word = current_word[6:]
      current_emotion = splitwords[5]
      current_emotion = current_emotion[14:]
      wordAndEmotion = {current_word : current_emotion}
      wordToScore.append(wordAndEmotion)
   return wordToScore
```

F Code Example 6 [Favorite Count, Retweet count]

```
#@author: Tucker Massad
#Returns average favorite count given a list of tweet objects
#@param tweetSet: List of tweet objects
def getAverageFavoriteCount(tweetSet):
   favCount = 0
   for tweet in tweetSet:
      favCount += tweet.get("favorite_count")
   return favCount/len(tweetSet)


#Returns average retweet count given a list of tweet objects
#@param tweetSet: List of tweet objects
def getAverageRetweetCount(tweetSet):
   retweetCount = 0
   for tweet in tweetSet:
```

```
        retweetCount += tweet.get("retweet_count")
    return retweetCount/len(tweetSet)
```

G Code Example 7 [Streaming Collection]

```
#@author: Tucker Massad
#Python script that places Trumps tweets and all replies to those tweets in a first
    database and all tweets mentioning Trump by name into a second database
from urllib3.exceptions import ProtocolError

def main():
  auth = OAuthHandler(settings.consumer_key,
  settings.consumer_secret)

  auth.set_access_token(settings.access_token,
  settings.access_secret)

  server = couchdb.Server("http://149.106.57.198:5984/")
  db = server[settings.database]
  talkdb = server[settings_2.database]

  #Listening for tweets coming from Trump
  trumplistener = CouchDBStreamListener(db)
  trumpstream = Stream(auth, trumplistener)

  #Listening for tweets that include keywords 'donald trump', 'trump', and 'Donald
      Trump'.
  talklistener = CouchDBStreamListener(talkdb)
  talkstream = Stream(auth, talklistener)


  startTime = time.time()
```

```
while(True):
  try:
    print("Streaming beginning!")
    trumpstream.filter(follow=['25073877'])
    talkstream.filter(track=['donald trump', 'trump', 'Trump', 'Donald Trump'])
  except ProtocolError:
    print("Protocol Error! Attempting to continue...")
    continue
  except IOError, ex:
    print 'Exception caught: %s' % ex
    continue
  except AttributeError:
    print 'Attribute error caught! Skipping tweet and continuing'
    continue
  except KeyboardInterrupt:
    endTime = time.time()
    print("Total tweets received: %d" % trumplistener.tweet_count)
    print("Total time elapsed (in seconds): ", endTime-startTime)
    print("Tweets per second: " , trumplistener.tweet_count/(endTime-startTime))

#Run
if __name__ == "__main__":
  main()
```

H Code Example 8 [Reply Count]

```
#@author Tucker Massad
#returns the quantity of replies a tweet has recieved in the past 24 hours
#@param tweet: Tweet object of desired tweet we would like count public replies made
    towards
def getReplyCount(tweet):
  replyID = tweet.get("in_reply_to_status_id")
  timeCreated = tweet.get("created_at")[:19]
  nextDayTime = tweet.get("created_at")[12:19]
```

```
  nextDayTime = nextDayTime.replace(":", "")
  replyCount = 0
  for possibleReply in dbTwo:
    if possibleReply.get("created_at")[:11] == timeCreated[:11] or
        possibleReply.get("created_at").replace(":", "") <= nextDayTime:
      if replyID == tweet.get("id_str"):
        replyCount+=1
  return replyCount
```

I Code Example 9 [Followers Gained]

```
#@author Tucker Massad
#returns estimation of amount of followers gained in 24-hour period after a tweet is
    sent
#@param tweet: Tweet object of desired tweet we would like to count the amount of
    followers gained 24-hours thereafter
#@param followingTweet: Tweet object of proceeding tweet we would like to use to
    judge the amount of followers gained [MUST BE SENT AT LEAST 24 HOURS AFTER tweet]
def followersGained(tweet, followingTweet):
  initialCount = tweet.get("user").get("follower_count")
  latterCount = followingTweet.get("user").get("follower_count")
  difference = abs(latterCount-initialCount)
  timeBetween = tweet.get("created_at")[11:19]-followingTweet.get("created_at")[11:19]
  followersPerHour = difference/(24+timeBetween)
  followersInDay = followersPerHour*24
  return("Followers gained 24 hours after sending tweet = " + str(followersInDay) +
      ". Followers-per-hour rate: " + str(followersPerHour))
```

J Code Example 10 [Public Mentions]

```
#@author Tucker Massad
#returns the amount of public tweets sent mentioning Trump by name 24-hours after
    Trump has tweeted
```

```
#@param tweet: Tweet object of desired tweet we would like to count the amount of
    public mentons are made in 24-hour period after tweet is sent
def publicMentions(tweet):
  date = tweet.get("created_at")[:19]
  exactDay = int(date[7:9])
  nextDay = exactDay + 1
  twentyFourAfter = date.replace(exactDay, nextDay)
  mentionCount = 0
  for possibleMention in dbThree:
    if possibleMention.get("id_str") not in idList:
      if possibleMention.get("created_at")[:11] == date[:11]:
        idList.append(possibleMention.get("id_str"))
        mentionCount += 1
      elif possibleMention.get("created_at")[7:9] == nextDay and
          possibleMention.get("created_at").replace(":", "") <= twentyFourAfter:
        idList.append(possibleMention.get("id_str"))
        mentionCount += 1
  return mentionCount
```

References

[1] Li Bing and Keith Chan. "A Fuzzy Logic Approach for Opinion Mining on Large Scale Twitter Data". In: *Journal of Big Data* (2015).

[2] bpb27. *twitter scraping*. https://github.com/bpb27/twitter_scraping. 2016.

[3] Paola Cerchiello and Paolo Giudici. "Big Data Analysis for Financial Risk Management". In: *Journal of Big Data* (2016).

[4] Kiran G'Arimella. "Quantifying Controversy in Social Media". In: *ACM Digital Library* (2016).

[5] Yuheng Hu, Shelly Farnham, and Kartik Talamadupula. "Predicting User Engagement on Twitter with Real-World Events". In: *AAAI Conference on Web and Social Media* (2016).

[6] Jason Huggins. *A browser automation framework and ecosystem*. https://github.com/SeleniumHQ/selenium. 2004.

[7] R. Junco, G. Heibergert, and E. Loken. “The effect of Twitter on college student engagement and grades”. In: *Journal of Big Data* (2016).

[8] Sunghwhan Mac Kim and Kyo Kaguera. “Twitter Content Eliciting User Engagement: A Case Study on Australian Organisations”. In: *Journal of Big Data* (2016).

[9] Tariq Mahmood. “Mining Twitter Big Data to Predict 2013 Pakisatan Election Winner”. In: *IEEE* (2014).

[10] Priya Nambisan. “Ruminating Behavior of Depression Revealed through Twitter”. In: *IEEE* (2015).

[11] Ekaternia Olshannikova, Jukka Huhtamaki, and Hannu Karkkainen. “Conceptualizing Big Social Data”. In: *Journal of Big Data* Vol. 4 (2016).

[12] B Padmaja and V. V. Rama Prasad. “TreeNet Analysis of Human Stress Behavor using Socio-Mobile Data”. In: *Journal of Big Data* Vol. 3 (2016).

[13] Joshua Roesslein. *Tweepy - Twitter library for python*. https://github.com/tweepy/tweepy. 2009.

[14] Koichi Sato. “Detecting Real-time Events using Twitter”. In: *IEEE* (2016).

[15] Uthayanath Suthakar et al. “An efficient strategy for the collection and storage of large volumes of data for computation”. In: *Journal of Big Data* (2016).

[16] Jason Urgo. *RealDonaldTrump Twitter Stats Summary Profile (Social Blade Twitter Statistics)*. SocialBlade.com, Social Blade LLC. 2017.

[17] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. *Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis*. 2005.

[18] Ruan Yefeng, Lina Alfatoukh, and Arjan Durresi. “Exploring Stock Market Using Twitter Trust Network”. In: *IEEE* (2016).