John Halloran

For this progress update, this project only used President Donald Trump's tweets[1] as a way to explore the different techniques to analyzing text. As of March 9th, the data has gone through some light cleaning to standardize some of the data such as the dates, preprocess the tweets' text to tokenize the data, and performed a word count. The instructions followed come from Marco Bonzanini[2] who provides a wonderful explanation on using Python to analyze text from Twitter. All of this work can be found on my personal GitHub repository[4].

The first part has to do with reorganizing the data on dates and time. This is for future plans when the project will interact more as a comparison with a data set on Hillary Clinton's tweets[3]. It is best to make comparisons on dates when there is a standard used. Donald Trump's data originally displayed in the data format YY-MM-DD. Instead we want the dates to look like MM/DD/YYYY. This only required a simple brute force method and some simple string manipulation [1]

```
#Cleaning date data - want to convert from YY-MM-DD to MM/DD/YYYY
numRows = len(donald_tweets['Date'])
count = 0
while (count < numRows):
    tmp = donald_tweets['Date'][count]
    tmp = tmp[3:5] + "/" + tmp[6:] + "/20" + tmp[0:2]
    #print(tmp)
    #print(donald_tweets['Date'][count])
    donald_tweets.set_value(count, 'Date', tmp)
    count = count + 1
donald_tweets.head(5)
```

Figure 1: Code to convert YY-MM-DD to MM/DD/YYYY

This code positively changed the data so that we are now dealing with the month followed by day. There is no need to change the time format since the standard should be based on the 24 hour clock to differentiate between morning and afternoon. This eliminates any future data

processing with dates and times to ignore any AM or PM strings. This same technique will also be applied to Hillary Clinton's data set as well.

The next portion to begin analyzing tweets is creating tokens. A token is defined to be a term in the tweet. For example, "We need leaders who can negotiate great deals for Americans. It is common sense. Let's Make America Great Again! https://t.co/u25yI5T7E8," will be separate all these words into separate tokens by the function preprocess. Preprocess will break the tweet down to tokens and return a list of them. These tokens are built on a defined set of regular expressions for strings in **regex_str**. These regular expressions are given by Bonzanini on his website and appear as such [1]

```
regex_str = [
    emoticons_str,
    r'<[^>]+>', # HTML tags
    r'(?:@[\w_]+)', # @-mentions
    r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)", # hash-tags
    r'http[s]?://(?:[a-z]|[0-9]|[$-_@.&amp;+]|[!*\(\),]|(?:%[0-9a-f][0-9a-f]))+', #
URLs

    r'(?:(?:\d+,?)+(?:\.?\d+)?)', # numbers
    r"(?:[a-z][a-z'\-_]+[a-z])", # words with - and '
    r'(?:[\w_]+)', # other words
    r'(?:\S)' # anything else
]
```

Figure 2: Regular Expressions

This is made possible mainly by the **nltk** library's **tokenize** package for **word_tokenize** in addition to the **re** package. **Word_tokenize** have a function called tokenize that does all of the token creation.

The advantage to using tokens, is that they create an easier way to count the most common words. Python also provides a **Counter** in the **collections** library to make a dictionary for counting word frequency. The algorithm to count is just a brute force method going through

each tweet and passing it off to preprocess, to get the tokens. After which, the counter will take those tokens and update its list. The code in general looks like[1]

```
numRows = len(donald_tweets['Tweet_Text'])
count = 0
count_all = Counter()
while (count < numRows):
    terms_all = [term for term in preprocess(donald_tweets['Tweet_Text'][count])]
    count_all.update(terms_all)
    count = count + 1
print(count_all.most_common(5))
```

Figure 3: Count up most common words among all tweets

This code, however, leads to very meaningless data as the results of this produce the following output, "[('.', 7690), ('!', 5189), ('"', 4695), (',', 3716), (':', 3484)]." This very meaningless, so we add what are called stop words. The algorithm will maintain a list of words that are common, but offer no real information such as "the," "or," "a," etc. Punctuation is also ignored. This is filtered at the line where **terms_all** is declared, as above. By adding a conditional, we can filter the terms.

```
terms = [term for term in preprocess(donald_tweets['Tweet_Text'][count]) if (term not
in stop and len(term) > 3 and "@" not in term and "#" not in term)]
```

Figure 4: Modified tokens that are accepted

This line in particular filters out all words less than 3 characters, any token that includes "@" or "#". The results from this are [('Trump', 1002), ('Thank', 754), ('great', 555), ('Hillary', 537), ('people', 377)]. Similarly, we can remove the "@" and "#" from the conditionals and figure out Trump's favorite mentions and hashtags, each being [('@FoxNews', 272), ('@CNN', 231), ('@megynkelly', 107), ('@foxandfriends', 100), ('@DanScavino', 90)] and [('#Trump2016', 592), ('#MakeAmericaGreatAgain', 508), ('#MAGA', 104), ('#DrainTheSwamp', 78), ('#AmericaFirst', 77)] respectively.

From this information, it is possible to apply these similar techniques to Hiliary Clinton's tweets. The most important part of setting up this data is the token generator. This will make it easier to break the tweets down word by word. The direction to maybe take this data is to correlate popular words with what tweets get the most favorites and retweets, figure out when hashtags become their most popular, and what might be the emotional values of these tweets might be.

Works Cited

1. Alidina, Abbas. *Donald Trump's 11 Best Tweets (and 7,000 Losers)*. Retrieved February 14, 2017 from Crowdbabble: https://www.crowdbabble.com/blog/the-11-best-tweets-of-all-time-by-donald-trump/.
2. Bonzanini, M. *Mining Twitter Data with Python*. Retrieved March 8th, 2017: https://marcobonzanini.com/2015/03/02/mining-twitter-data-with-python-part-1/.
3. Ginns, Andrew. *Idp-twitter*. Retrieved February 14, 2017 from GitHub: https://github.com/andrewginns/idp-twitter/blob/master/HillaryClinton_tweets.csv.
4. Halloran, John. *DataScience_Marquette*. Retrieved March 9, 2017 from GitHub: https://github.com/johnnyguitar95/DataScience_Marquette/tree/master/Notebooks.