BUSINESS INTELLIGENCE WAC

Submitted to Ussama Yaqub

19010010, 19010050

Business Intelligence

Introduction

For us, the objective of this project was to learn the tools of the trade for data analytics or at least make ourselves familiar. Part of the primary objective was not to fall back on using Microsoft Excel. During our discussions with our instructor, we realized the sheer computational and analytical power of Python. We also appreciated the scalability and customization offered by Python scripts. Consequentially, our primary task was to learn coding on Python and use MySQL DB management purposes.

This project was more of a challenge to complete and one of the most meaningful projects, in terms of learning, we have done in our MBA. The process we employed was mostly based on trial and error; diagnosing syntax errors on Python and correcting them one line at a time. We referred to countless tutorials on YouTube and discussion forums on the website Stack Overflow.

Process / Methodology

We decided that we will open the data file, which was TXT in MySQL to convert it into CSV format to be used in Python. However, we later realized that we did not need to do that as Python can also read TXT files. We also used MySQL for some exploratory data analysis initially.

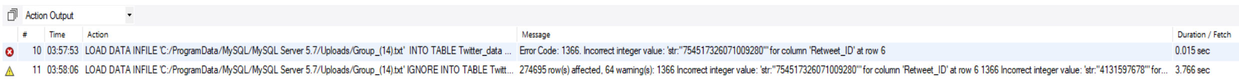
Once the data was converted into CSV, we imported CSV into Python and started our data analytics. We used Python to do most of our descriptive data analyses: Time Series Analysis, Hypothesis Testing, and Regression Analysis and Correlational analysis. We also used Python for our visualization purposes.

Once the data was ready to be visualized, we used Power BI’s python integration module to import our visualization into Power BI. The reason behind this method instead of linking MySQL database to Power BI was that this would make our agile, as any change in the source TXT file can easily be updated in Power BI compared to connecting MYSQL database, where the table needs to be deleted and made again to be updated.

Execution

MySQL

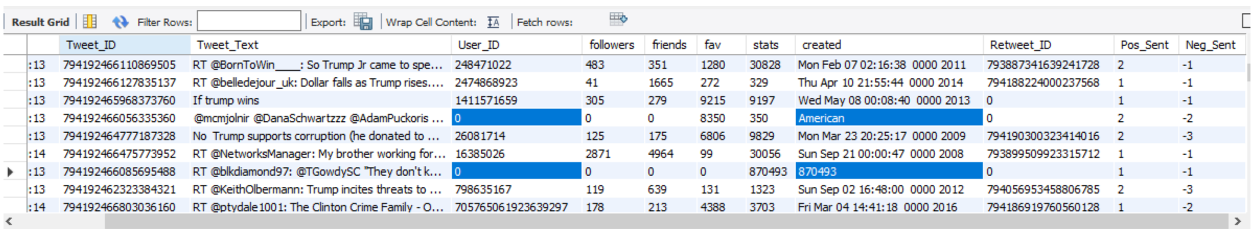
We started with opening our text file in MySQL, only to realize that there was a Workbench version was not compatible with the code provided.A screenshot of the error message is shown below:



#	Time	Action	Message	Duration / Fetch
10	03:57:53	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/Group_(14).txt' INTO TABLE Twitter_data ...	Error Code: 1366. Incorrect integer value: 'str:"754517326071009280"' for column 'Retweet_ID' at row 6	0.015 sec
11	03:58:06	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/Group_(14).txt' IGNORE INTO TABLE Twitt...	274695 row(s) affected, 64 warning(s): 1366 Incorrect integer value: 'str:"754517326071009280"' for column 'Retweet_ID' at row 6 1366 Incorrect integer value: 'str:"4131597678"' for...	3.766 sec

incorrect data importing

Thanks to Google, YouTube & StackOverflow, we tweaked the code to meet our requirements. We succeeded in getting the file loaded into MySQL with some errors in the data file as shown in the picture below:



	Tweet_ID	Tweet_Text	User_ID	followers	friends	fav	stats	created	Retweet_ID	Pos_Sent	Neg_Sent
:13	794192466110869505	RT @BornToWin____: So Trump Jr came to spe...	248471022	483	351	1280	30828	Mon Feb 07 02:16:38 0000 2011	793887341639241728	2	-1
:13	794192466127835137	RT @belledejour_uk: Dollar falls as Trump rises....	2474868923	41	1665	272	329	Thu Apr 10 21:55:44 0000 2014	794188224000237568	1	-1
:13	794192465968373760	If trump wins	1411571659	305	279	9215	9197	Wed May 08 00:08:40 0000 2013	0	1	-1
:13	794192466056335360	@mcmjolnir @DanaSchwartzzz @AdamPuckoris ...	0	0	0	8350	350	American	0	2	-2
:13	794192464777187328	No Trump supports corruption (he donated to ...	26081714	125	175	6806	9829	Mon Mar 23 20:25:17 0000 2009	794190300323414016	2	-3
:14	794192466475773952	RT @NetworksManager: My brother working for...	16385026	2871	4964	99	30056	Sun Sep 21 00:00:47 0000 2008	793899509923315712	1	-1
:13	794192466085695488	RT @blkdiamond97: @TGowdySC "They don't k...	0	0	0	0	870493	870493	0	1	-1
:13	794192462323384321	RT @KeithOlbermann: Trump incites threats to ...	798635167	119	639	131	1323	Sun Sep 02 16:48:00 0000 2012	794056953458806785	2	-3
:14	794192466803036160	RT @ptydale1001: The Clinton Crime Family - O...	705765061923639297	178	213	4388	3703	Fri Mar 04 14:41:18 0000 2016	794186919760560128	1	-2

Errors in the fields, this is in SQL by importing the txt file initially.

Cause of the Problem

This problem was arising from the fact that MySQL and Python read the data with an understanding that each "Enter" keypress means new row of data, not realizing that there were 1300+ tweets where the person used "Enter" key in their tweets, resulting in string data occupying numeric fields and many other fields being populated with inconsistent data. This caused distortion of the data. We now had to remove the distorted data.

Once the data was opened in MySQL, we now had to add another column into the table which will include the average (mean) of the Positive and Negative Sentiment values (Avg_Sent). Our first step was to create another table in the database which will include the column for the new calculated variable, so we used the same syntax provided by our instructor and added another column heading in the code. We used the following code to execute our query and save the output as CSV to be imported to Python.

```sql
SELECT Tweet_Date, Tweet_ID, Tweet_Text, User_ID, followers, friends, fav, stats, created, Retweet_ID, Pos_Sent, Neg_Sent , (Pos_Sent + Neg_Sent) / (2) AS Average
FROM twitter_data
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/Projfile.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

We used this new ProjfileCSV as our core data file in Python to extract and analyze the data. Once we had a CSV file, we cleansed the data again to remove the above-mentioned distorted data.

We loaded this data file into the new table we created with the **Avg_Sent** column and ran some basic queries to get the feel of the data.

Now that our ProjfileCSV was ready, we switched to Python for our data analytics.

PYTHON

In python, we started by loading the following libraries:

- Pandas
- Numpy
- Matplotlib
- Seaborn
- WorldCloud
- Statsmodels.api

```python
In [2]:  import pandas as pd
         import numpy as np
         import matplotlib as mpl
         import matplotlib.pyplot as plt
         import seaborn as sns
         from subprocess import check_output
         from wordcloud import WordCloud, STOPWORDS
         import statsmodels.api as sm
         %matplotlib inline
```

Figure 1 code for importing libraries

Our working in python is divided into seven distinct sections

Data Import

In this section, we loaded our libraries and imported our projfileCSV into the python. We imported the data without headers, so we defined header as column_names and assigned those as our headers. We replaced default index with "Tweet_Date" and set it as datetimeindex, and we also ran some diagnostics such as missing data per column, types of each column available in the data and set the data type of "stats" column as a float, so that we can use it for calculations in the future.

```
In [3]:  #Importing file into Python
         df = pd.read_csv('C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/Projfile.csv', header=None, parse_dates=True, )

In [5]:  #Checking df
         #df.head()

In [6]:  #Defining Column Names as the data dont have Column names
         column_names = ['Tweet_Date','Tweet_ID','Tweet_Text','User_ID','followers','friends','fav','stats','created','Retweet_ID','Po

In [7]:  ##Defining Column Labels Just in case
         column_Labels = ['Tweet_Date','Tweet_ID','Tweet_Text','User_ID','followers','friends','fav','stats','created','Retweet_ID','P

In [8]:  # Replacing default column names with the ones we created
         df.columns =column_names

In [9]:  # Replacing default Index with Tweet_Date
         df.set_index('Tweet_Date', inplace=True)

In [10]: # Converting INdex to Datetimeindex ( for timeseries)
         df.index = pd.to_datetime(df.index)

In [11]: # Converting stats column to flot from string format
         df['stats'] = df['stats'].astype(float)
```

Figure 2 importing csv file into Python

Setting up Sub-Data Frames for Analysis

In this section, we created sub-data frames as per our needs such as we creating a data frame where it would show us "Tweet_Text" and "Avg_Sent" (average sentiment) column if the "Tweet_Text" included the term "Trump" in its string. When we did the same for Clinton, we noticed that the results were almost identical for Hilary and Clinton if tested separately.

Moreover, we acknowledged instances where both Trump and Clinton would be mentioned within the same tweet. Including sentiments from such tweets would seemingly distort our analysis as the sentiment cannot be linked to a particular candidate. For this, we excluded instances where both names were mentioned as shown in the code:

```
In [95]:  # Display Test for Filtered Tweets
          #df[df['Tweet_Text'].str.contains('Trump')][["Tweet_Text","Avg_Sent"]].head()

In [96]:  # Filtering Trump in tweets and showing tweets and average_sentiment
          df_Trump = df[df['Tweet_Text'].str.contains('Trump')][["Tweet_Text","Avg_Sent"]]

In [97]:  #print(df_Trump)
          #df_Trump.head()

In [148]: #Tweets with Trump but no mention of Clinton
          T_minus_C = df_Trump[~df_Trump["Tweet_Text"].str.contains("Clinton")][["Tweet_Text","Avg_Sent"]]

In [161]: #print(T_minus_C.describe())

In [149]: # Filtering Clinton in tweets and showing tweets and average_sentiment
          df_Clinton = df[df['Tweet_Text'].str.contains('Clinton')][["Tweet_Text","Avg_Sent"]]

In [136]: #print(df_Clinton)
          #df_Clinton.head()

In [150]: #Tweets with Clinton but no mention of Trump
          C_minus_T = df_Clinton[~df_Clinton["Tweet_Text"].str.contains("Trump")][["Tweet_Text","Avg_Sent"]]

In [162]: #print(C_minus_T.describe())
```

Figure 3 In lines 148 and 150, we exclude instances where both Trump and Clinton are mentioned

Descriptive Data Analysis

In this section, we calculated summary statistics for columns/variables, calculated average sentiment and furthermore found out sentiments for Trump and Clinton.

Note: We were not able to calculate one figure for sentiment for Trump and Clinton, so we used MySQL for that.

Descriptive Data Analysis

```
In [267]: # Descriptive Summary Statistics for numercial columns in the table

          print(df.describe())

                Tweet_ID       User_ID     followers        friends           fav  \
count   2.693910e+05  2.693910e+05  2.693910e+05  269391.000000  2.693910e+05
mean    7.940000e+17  1.318122e+17  7.724916e+03    1889.485406  3.869961e+05
std     3.328774e+06  2.862533e+17  1.966643e+05    7821.701458  2.836272e+07
min     7.940000e+17  0.000000e+00  0.000000e+00       0.000000  0.000000e+00
25%     7.940000e+17  1.805237e+08  1.370000e+02     179.000000  4.780000e+02
50%     7.940000e+17  9.775224e+08  4.790000e+02     523.000000  3.413000e+03
75%     7.940000e+17  3.410690e+09  1.581000e+03    1716.000000  1.265850e+04
max     7.940000e+17  7.940000e+17  3.118808e+07  762898.000000  2.147484e+09

                   stats    Retweet_ID       Pos_Sent       Neg_Sent       Avg_Sent
count   2.693910e+05  2.693910e+05  269391.000000  269391.000000  269391.000000
mean    5.557376e+05  5.873656e+17       1.339507      -1.678605      -0.169549
std     3.333890e+07  3.479573e+17       0.616636       0.925200       0.550954
min     0.000000e+00  0.000000e+00       1.000000      -5.000000      -2.000000
25%     3.057000e+03  0.000000e+00       1.000000      -2.000000      -0.500000
50%     1.160100e+04  7.940000e+17       1.000000      -1.000000       0.000000
75%     3.548050e+04  7.940000e+17       2.000000      -1.000000       0.000000
max     2.147484e+09  7.940000e+17       5.000000      -1.000000       2.000000
```

Figure 4 Calculating summary statistics for descriptive analysis

```
In [154]: # Average Sentiment About Trump

          ##Used MySQL Query here in MYSQL SELECT AVG(Avg_Sent) FROM twitter_data_2 WHERE Tweet_Text LIKE '%Trump%'
          #resulting in -0.1875 value for Avg_Sent

          #python syntex
          T_minus_C.Avg_Sent.mean()

Out[154]: -0.16430071023361772

In [153]: # Average Sentiment About Clinton

          ##Used MySQL Query here in MYSQL SELECT AVG(Avg_Sent) FROM twitter_data_2 WHERE Tweet_Text LIKE '%Clinton%'
          #resulting in -0.2766 value for Avg_Sent

          #python syntex
          #C_minus_T.Avg_Sent.mean()

Out[153]: -0.24540291036611867
```

Figure 5 Calculating Avg_Sent for Trump and Clinton. Note that there we have incorporated exclusion of instances where both names have been mentioned

In this section we also calculated 100 most common words used in the tweets, 100 most common words associated with Trump and 100 most common words associated with Clinton as well.

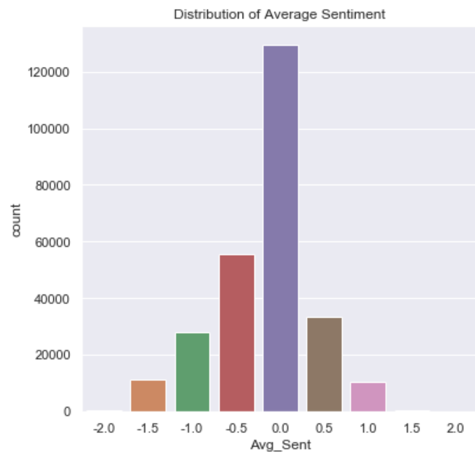
```
In [274]: # 100 Most common words used in tweets with Hillary Clinton

from subprocess import check_output    ##Can Delete
from wordcloud import WordCloud, STOPWORDS  ##Can Delete

mpl.rcParams['figure.figsize']=(8.0,6.0)    #(6.0,4.0)
mpl.rcParams['font.size']=12                #10
mpl.rcParams['savefig.dpi']=100             #72
mpl.rcParams['figure.subplot.bottom']=.1


stopwords = set(STOPWORDS)

wordcloud = WordCloud(
                          background_color='white',
                          stopwords=stopwords,
                          max_words=100,
                          max_font_size=60,
                          random_state=42
                         ).generate(str(C_minus_T['Tweet_Text']))

print(wordcloud)
fig = plt.figure(1)
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
fig.savefig("word1.png", dpi=900)

<wordcloud.wordcloud.WordCloud object at 0x0000021AC76CB630>
```

Figure 6 The WordCloud Module Output for 100 most repeated words for str.constraint = Clinton

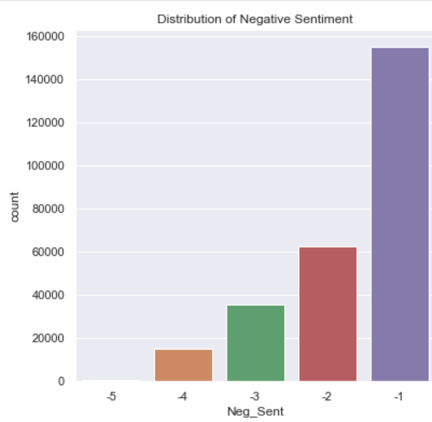
Sentiment Distribution

We performed some descriptive analysis for gauging the distribution of sentiments across amongst the population of the users in the data. We obtained the following graphs in Jupyter IDE. We found that Positive ratings of 1 and Negative ratings of -1 were most common. Also, an average sentiment rating of 0 was most common. The distribution was not perfectly normal as it was skewed to the left (negative sentiment), which showed that people were generally more skeptical than appreciative.

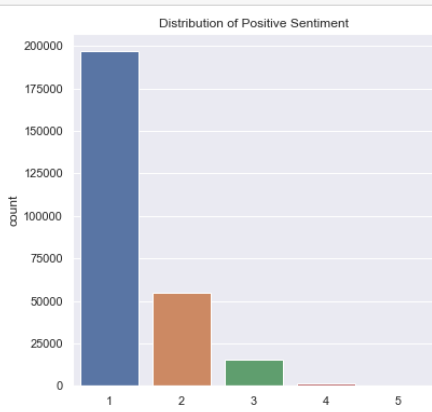
In [254]:
```
#Distribution of Average Sentiment ( Once Column at a time)
sns.set(rc={'figure.figsize':(6,6)})
plt.title('Distribution of Average Sentiment')
sns.countplot(x = 'Avg_Sent', data = df);
```



In [256]:
```
#Distribution of Negative Sentiment ( Once Column at a time)
sns.set(rc={'figure.figsize':(6,6)})
plt.title('Distribution of Negative Sentiment')
sns.countplot(x = 'Neg_Sent', data = df);
```



In [255]:
```
#Distribution of Positive Sentiment ( Once Column at a time)
sns.set(rc={'figure.figsize':(6,6)})
plt.title('Distribution of Positive Sentiment')
sns.countplot(x = 'Pos_Sent', data = df);
```



Overall Top 100 Words

```python
# 100 Most common words used in All tweets

from subprocess import check_output ##Can Delete
from wordcloud import WordCloud, STOPWORDS  ##Can Delete

mpl.rcParams['figure.figsize']=(8.0,6.0)    #(6.0,4.0)
mpl.rcParams['font.size']=12                #10
mpl.rcParams['savefig.dpi']=100             #72
mpl.rcParams['figure.subplot.bottom']=.1


stopwords = set(STOPWORDS)

wordcloud = WordCloud(
                          background_color='white',
                          stopwords=stopwords,
                          max_words=100,
                          max_font_size=60,
                          random_state=42
                         ).generate(str(df['Tweet_Text']))

print(wordcloud)
fig = plt.figure(1)
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
fig.savefig("word1.png", dpi=900)
```

```
<wordcloud.wordcloud.WordCloud object at 0x0000021ABCD37198>
```

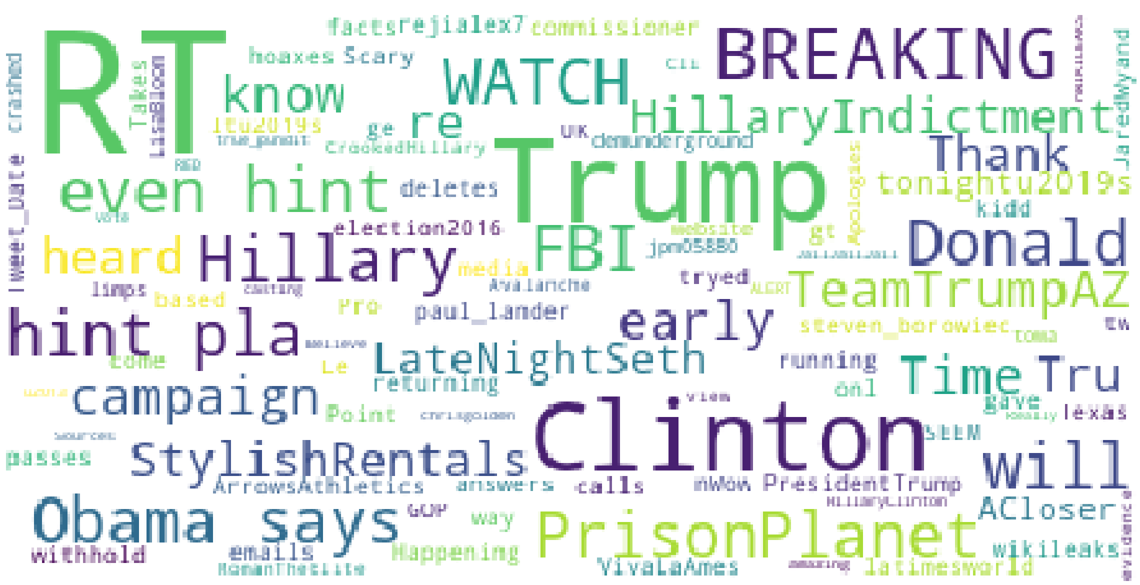


Figure 7 Top 100 Words WordCloud Output for Entire Data

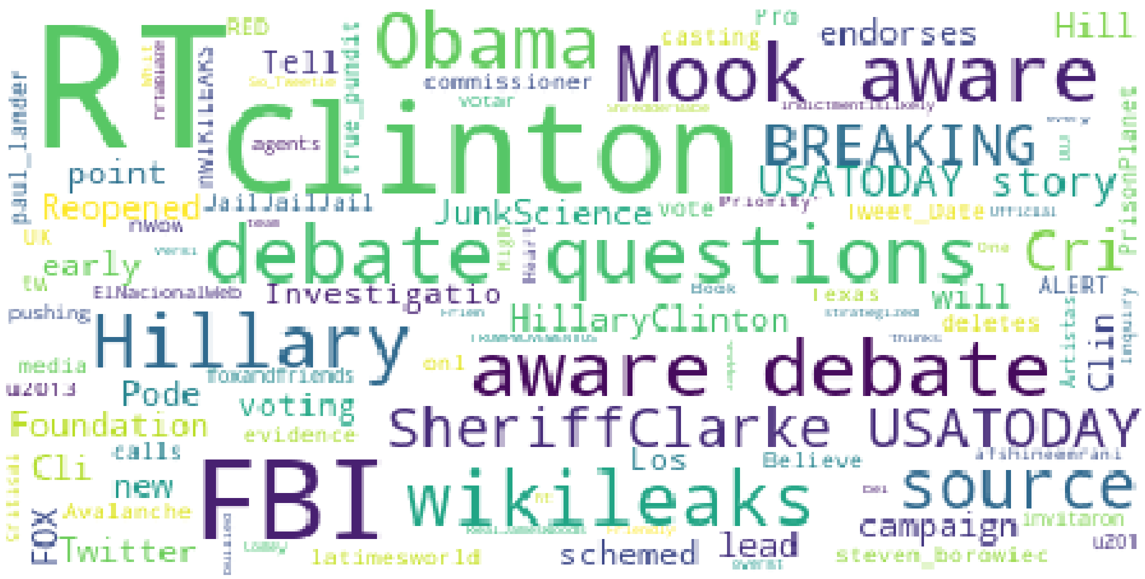
Clinton Top 100 Words

```
# 100 Most common words used in tweets with Hillary Clinton

from subprocess import check_output    ##Can Delete
from wordcloud import WordCloud, STOPWORDS  ##Can Delete

mpl.rcParams['figure.figsize']=(8.0,6.0)    #(6.0,4.0)
mpl.rcParams['font.size']=12                #10
mpl.rcParams['savefig.dpi']=100             #72
mpl.rcParams['figure.subplot.bottom']=.1


stopwords = set(STOPWORDS)

wordcloud = WordCloud(
                          background_color='white',
                          stopwords=stopwords,
                          max_words=100,
                          max_font_size=60,
                          random_state=42
                         ).generate(str(C_minus_T['Tweet_Text']))

print(wordcloud)
fig = plt.figure(1)
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
fig.savefig("word1.png", dpi=900)
```

```
<wordcloud.wordcloud.WordCloud object at 0x0000021AC76CB630>
```

When we executed the WordCloud library on python, we obtained the output reproduced below. The figure shows the words with their sizes in proportion to their occurrence. The RT shows the amount of re-tweets with the word ‘Clinton’ in the tweets. Also, we also see that the word ‘FBI’ appears much in tweets containing references to Hilary Clinton. Because of the incident regarding the FBI investigations into alleged acts of Clinton which compromised information relating to national security by accessing her email through un-secure networks. Also, the term ‘FBI’ would generally have a negative sentiment rating due to its association with crimes or criminal negligence.

We also see that ‘Sheriff Clarke’ also appears on the WordCloud output. On further investigation, we found that Sheriff Clarke is a former Milwaukee County Sherrif who made numerous appearances on national television in the wake of the Ferguson shootings and the associated backlash in the shape of the ***Black Lives Matter*** movement. Clarke lashed out on Clinton on national television for her remarks on Trump which he called racist. Being an African American, his remarks were picked up and much voiced on the Trump-supporting media channels like Fox News (we also see Fox appearing on the WordCloud output!). Clarke also slammed Clinton for comparing Trump to Harvey Weinstein in the wake of his leaked audio tapes which made headlines for their sexist remarks made by Trump himself.



Trump Top 100 Words

```
In [273]: # 100 Most common words used in tweets with Doland Trump

          from subprocess import check_output   ##Can Delete
          from wordcloud import WordCloud, STOPWORDS   ##Can Delete

          mpl.rcParams['figure.figsize']=(8.0,6.0)    #(6.0,4.0)
          mpl.rcParams['font.size']=12                #10
          mpl.rcParams['savefig.dpi']=100             #72
          mpl.rcParams['figure.subplot.bottom']=.1


          stopwords = set(STOPWORDS)

          wordcloud = WordCloud(
                                    background_color='white',
                                    stopwords=stopwords,
                                    max_words=100,
                                    max_font_size=60,
                                    random_state=42
                                   ).generate(str(T_minus_C['Tweet_Text']))

          print(wordcloud)
          fig = plt.figure(1)
          plt.imshow(wordcloud)
          plt.axis('off')
          plt.show()
          fig.savefig("word1.png", dpi=900)

          <wordcloud.wordcloud.WordCloud object at 0x0000021ABCD92B38>
```

Figure 8 The code for executing WordCloud library on identifying Top 100 words used with str.constraint="Trump"

The Top 100 WordCloud for Trump was very interesting. The word “StylishRentals” was significantly visible. According to our investigation, Stylish Rentals is a real estate service which offers commercial and residential accommodation for rent. The Trump Tower may have had some intersection with our data as these tweets may have something to do with the Trump Tower and not with Donald Trump’s presidential campaign. It may have been that the Trump Tower may have been involved in the public discussion during the course of the presidential campaign.

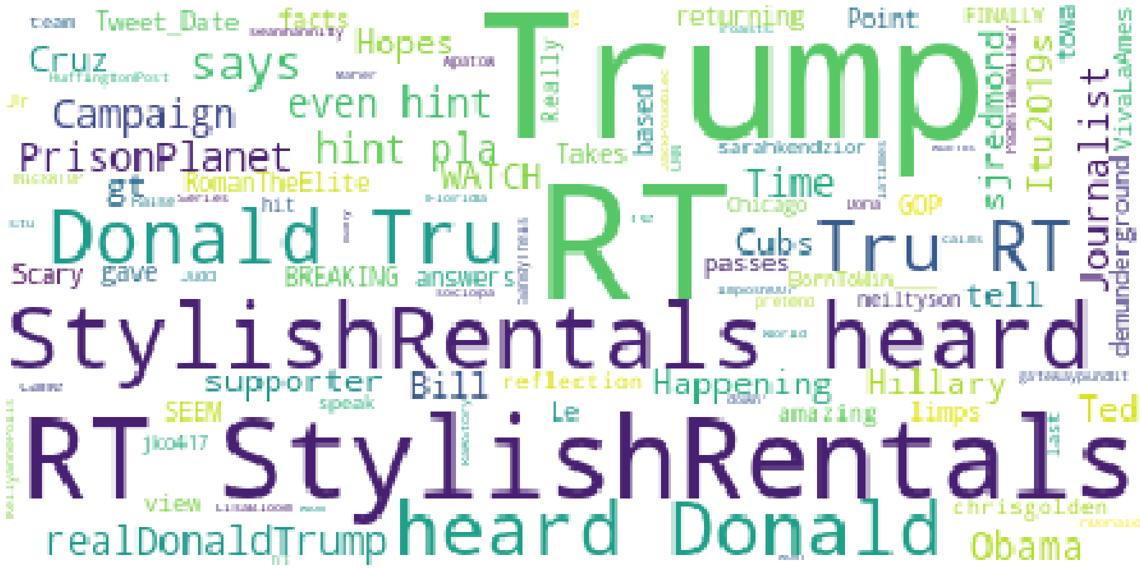


Figure 9 Top 100 WordCloud output for Trump

Time Series Analysis

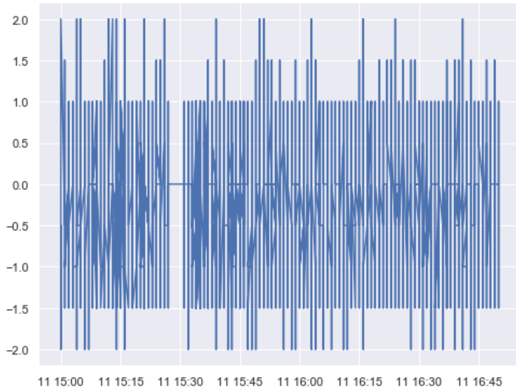
We created new data frames, just like the one created before with conditions such as Trump and Clinton in the String of “Tweet_Text” but only showing “Avg_Sent” column. We created these data frame for the purpose of plotting Avg_Sent against “Tweet_Date” to getting the plot of average sentiment over time.

The initial Time Series Analysis graphs were all over the place and did not provide us with much insights. We then decided to have times series for each candidate (Clinton and Trump). The resulting graphs still showed much fluctuation. We then had an hourly average sentiment graph for the candidates. However, the graph still showed much fluctuation and did not show a clear trend in data. We then further refined the output by incorporating a ***2-minute rolling average*** over the course of the duration. The result was a much more refined graph which demonstrated the trend of the average sentiment changing with time.

We resample the average sentiment data hourly and at two minutes periods and also calculated smoothed (rolling average) for the same data at two minutes intervals.

In [276]:
```
#PLot of Clinton Average Sentiment X=Tweet_Time y=Avg_Sent

df_plt_C = C_minus_T[["Avg_Sent"]]
plt.plot(df_plt_C,)
```

Out[276]: `[<matplotlib.lines.Line2D at 0x21abcdeb048>]`



In [277]:
```
#Hourly Average Sentiment in data
Hourly_D_Mean = df.resample('H').mean()
print(Hourly_D_Mean.Avg_Sent)
```

```
Tweet_Date
2016-03-11 15:00:00   -0.165826
2016-03-11 16:00:00   -0.173378
Freq: H, Name: Avg_Sent, dtype: float64
```

In [278]:
```
#Hourly Average Sentiment for Trump With HOurly avg_sent PLot
Hourly_T_Mean = T_minus_C.resample('H').mean()
print(Hourly_T_Mean)
plt.plot(Hourly_T_Mean)
```

```
                        Avg_Sent
Tweet_Date
2016-03-11 15:00:00 -0.154026
```

In [287]:
```
#2 mintues Average Sentiment for Clinton With 2 mintues avg_sent PLot ( can do 2 min rolling avg)
twomin_C_Mean = C_minus_T.resample('120S').mean()
#print(twomin_C_Mean)
plt.plot(twomin_C_Mean)

Smoothed_C_Mean = twomin_C_Mean.rolling(5).mean()
plt.plot(Smoothed_C_Mean)
```

Out[287]: `[<matplotlib.lines.Line2D at 0x21abcd0f550>]`

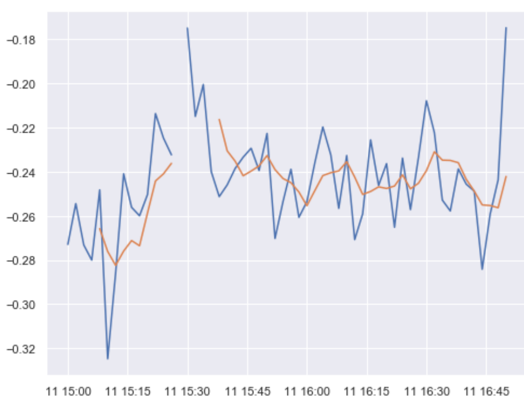
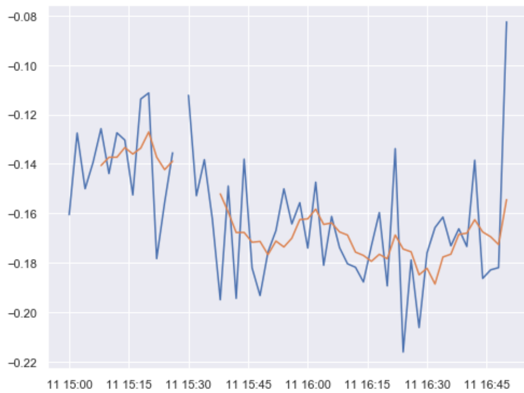


Figure 10 Line in Red shows the 2-minute Rolling Avg

In [286]:
```
#2 mintues Average Sentiment for Trump With 2 mintues avg_sent PLot ( can do 2 min rolling avg)
twomin_T_Mean = T_minus_C.resample('120S').mean()
#print(twomin_T_Mean)
plt.plot(twomin_T_Mean)

Smoothed_T_Mean = twomin_T_Mean.rolling(5).mean()
plt.plot(Smoothed_T_Mean)
```

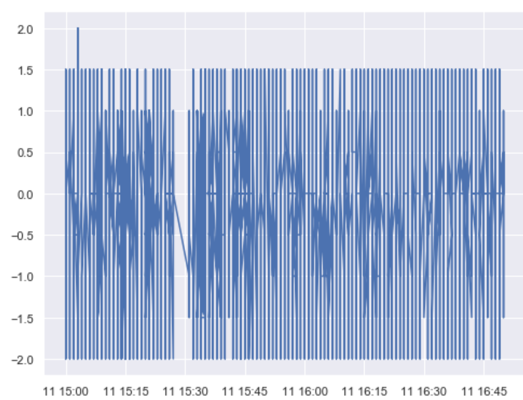
Out[286]: [<matplotlib.lines.Line2D at 0x21abcd54ef0>]



Start of Time Series Analysis for both Trump & Clinton

In [275]:
```
#Plot of Trump Average Sentiment X=Tweet_Time y=Avg_Sent

df_plt_T = T_minus_C[["Avg_Sent"]]
plt.plot(df_plt_T)
```

Out[275]: [<matplotlib.lines.Line2D at 0x21abce4af60>]



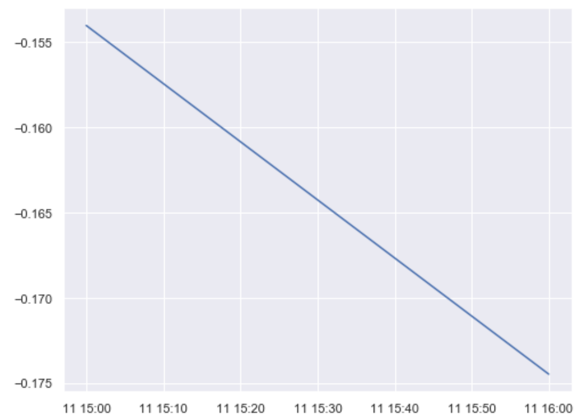
Hourly Sentiment Trends

The graphs below show that for Trump's tweets, the average sentiments decreased per hour while those for Clinton, increased (although with a negligible margin).

```
#Hourly Average Sentiment for Trump With HOurly avg_sent PLot
Hourly_T_Mean = T_minus_C.resample('H').mean()
print(Hourly_T_Mean)
plt.plot(Hourly_T_Mean)
```

```
                      Avg_Sent
Tweet_Date
2016-03-11 15:00:00 -0.154026
2016-03-11 16:00:00 -0.174495
```

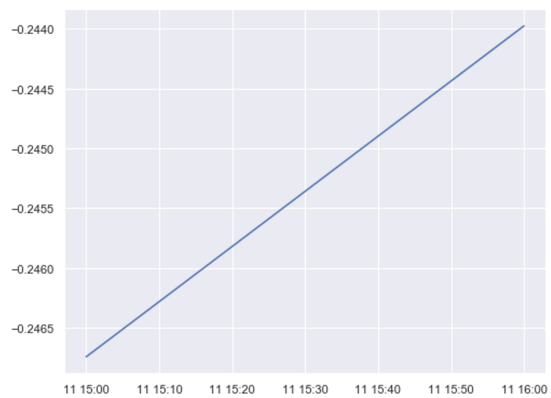
```
[<matplotlib.lines.Line2D at 0x21abccf42e8>]
```




```
#Hourly Average Sentiment for Clinton With HOurly avg_sent PLot
Hourly_C_Mean = C_minus_T.resample('H').mean()
print(Hourly_C_Mean)
plt.plot(Hourly_C_Mean)
```

```
                      Avg_Sent
Tweet_Date
2016-03-11 15:00:00 -0.246744
2016-03-11 16:00:00 -0.243976
```

```
[<matplotlib.lines.Line2D at 0x21ab7e69f98>]
```



Hypothesis Testing

```
# Our Hypothesis is that The average sentiment for both Clinton & Trump is NOT same and that there is a significant difference.
# H1 = Avgerage sentiment of Trump =/= Average sentiment of Clinton
# H0 = Average sentiment of Trump = Average sentiment of Clinton
# the P-Value of of the below Two-Sample T-Test shows is lower than 0.05 hence resulting is rejecting Null Hypothesis
import scipy.stats as sp
stats.ttest_ind(a= df_plt_T,
                b= df_plt_C,
                equal_var=False)    # Assume samples have equal variance?
```

```
Ttest_indResult(statistic=array([31.361325]), pvalue=array([3.01393444e-215]))
```

- Hypothesis Testing
- Regression Analysis

In this section, we wanted to see what kind of relationship exists among the different variables so we defined seven regression models to be tests, where X represents independent variables and Y represents dependent variables. The seven models are given below

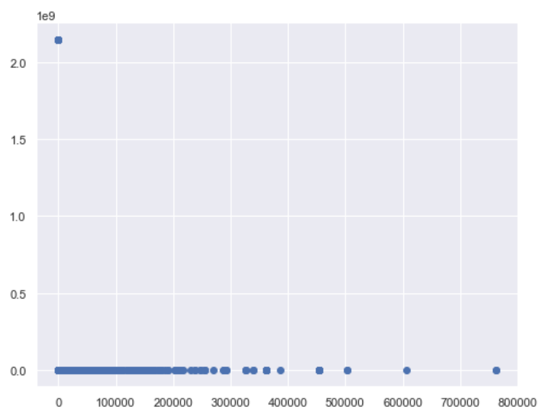
X	**Y**

Followers	**Avg_Sent**
Friends	**Avg_Sent**
Fav	**Avg_Sent**
Stats	**Avg_Sent**
Friends	**Fav**
Friends	**stats**
Fav	**stats**

We created scatter plot of the variables to get the general sense of the data before running regression model using statsmodels.api library. The purpose of using this library was it gave us the option to ignore the missing values in our columns.

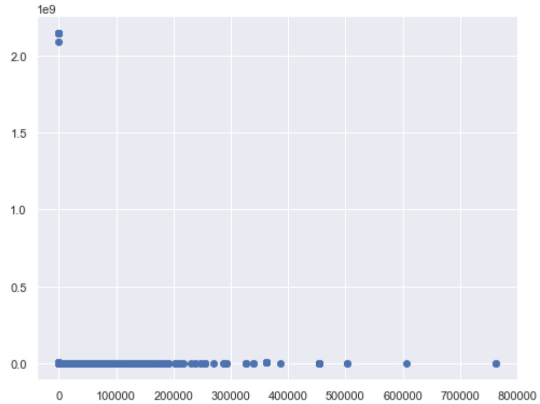
In [292]:
```
#      X=Friends       Y=Fav

plt.scatter(df.friends,df.fav)

x = df.friends
y= df.fav

model = sm.OLS(y, x, missing='drop')
results = model.fit()
results.params
```

Out[292]:
```
friends    0.843136
dtype: float64
```



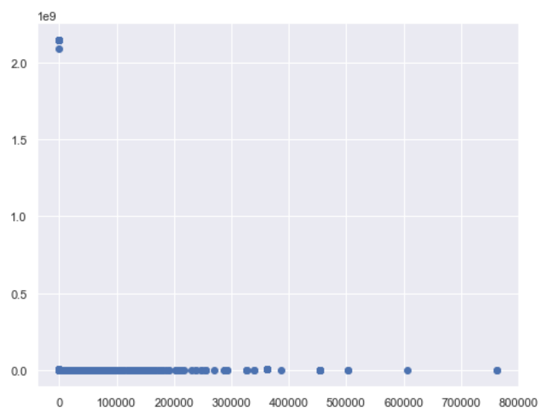
In [293]:
```python
#      X=Friends       Y=stats

plt.scatter(df.friends,df.stats)

x = df.friends
y= df.stats

model = sm.OLS(y, x, missing='drop')
results = model.fit()
results.params
```

Out[293]:
```
friends    4.895673
dtype: float64
```



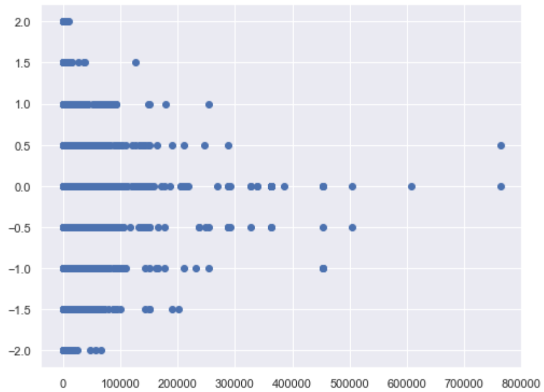
In [294]:
```python
#      X=Fav           Y=stats

plt.scatter(df.friends,df.stats)

x = df.fav
y= df.stats

model = sm.OLS(y, x, missing='drop')
results = model.fit()
results.params
```

Out[294]:
```
fav    0.999398
dtype: float64
```



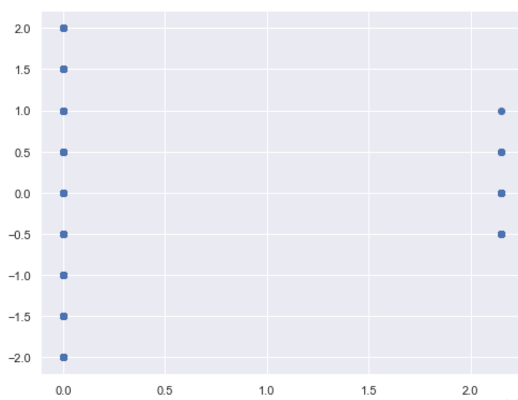
In [289]:
```
#      X= Friends      Y= Avg_Sent

plt.scatter(df.friends,df.Avg_Sent)

x = df.friends
y= df.Avg_Sent

model = sm.OLS(y, x, missing='drop')
results = model.fit()
results.params
```

Out[289]:
```
friends   -0.000005
dtype: float64
```

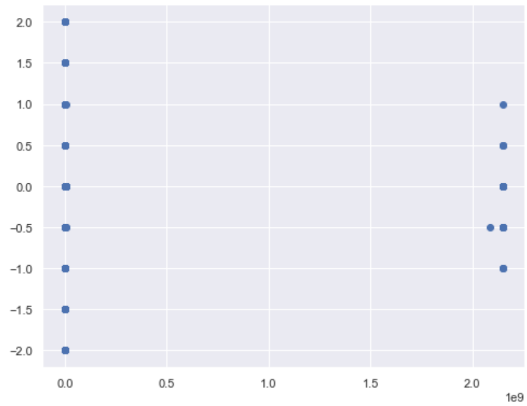


In [290]:
```
#      X=fav           Y=Avg_Sent

plt.scatter(df.fav,df.Avg_Sent)

x = df.fav
y= df.Avg_Sent

model = sm.OLS(y, x, missing='drop')
results = model.fit()
results.params
```

Out[290]:
```
fav    -7.801410e-12
dtype: float64
```



```
In [291]: #      X=stats          Y=Avg_Sent

          plt.scatter(df.stats,df.Avg_Sent)

          x = df.stats
          y= df.Avg_Sent

          model = sm.OLS(y, x, missing='drop')
          results = model.fit()
          results.params

Out[291]: stats   -3.049701e-11
          dtype: float64
```



Correlational Analysis

Lastly, we created another data frame for correlation purpose with the following columns, followers, friends, fav, stats, avg_sent. We created correlation matrix/table and also developed a heatmap visual for the correlation matrix. This was made possible by the seaborn module library in Python. The correlation graphs showed that there was significant correlation between the number of statuses and the statuses being favourited by users.

Correlational Analysis

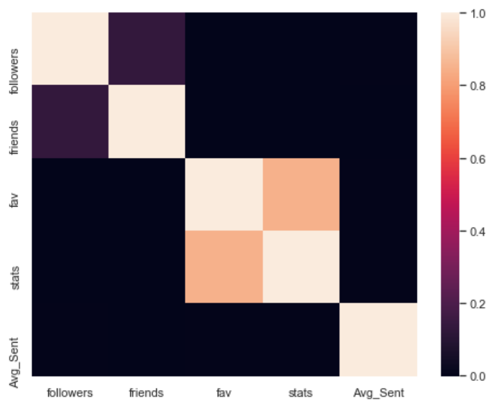
In [295]:
```
#Creating New sub-Dataframe  for Correlation section with relative variables

#Also includes Corr.Table and Heatmap

Corr_table = df[["followers","friends",'fav','stats','Avg_Sent'] ]

corr_matrix = Corr_table.corr().abs()

print(Corr_table.corr())

sns.heatmap(Corr_table.corr())
```

```
           followers   friends       fav     stats  Avg_Sent
followers   1.000000  0.130630 -0.000515 -0.000282  0.003888
friends     0.130630  1.000000 -0.003050 -0.002811 -0.002922
fav        -0.000515 -0.003050  1.000000  0.850158  0.003797
stats      -0.000282 -0.002811  0.850158  1.000000  0.003284
Avg_Sent    0.003888 -0.002922  0.003797  0.003284  1.000000
```

Out[295]: `<matplotlib.axes._subplots.AxesSubplot at 0x21ab4681be0>`


```
           followers   friends       fav     stats  Avg_Sent
followers   1.000000  0.130630 -0.000515 -0.000282  0.003888
friends     0.130630  1.000000 -0.003050 -0.002811 -0.002922
fav        -0.000515 -0.003050  1.000000  0.850158  0.003797
stats      -0.000282 -0.002811  0.850158  1.000000  0.003284
Avg_Sent    0.003888 -0.002922  0.003797  0.003284  1.000000
```

Out[295]: `<matplotlib.axes._subplots.AxesSubplot at 0x21ab4681be0>`



Power BI

We used Python integration feature in PowerBI to import some of our visuals into PowerBi while also using PowerBI own powerful visualization toolkits available to create our visualizations.

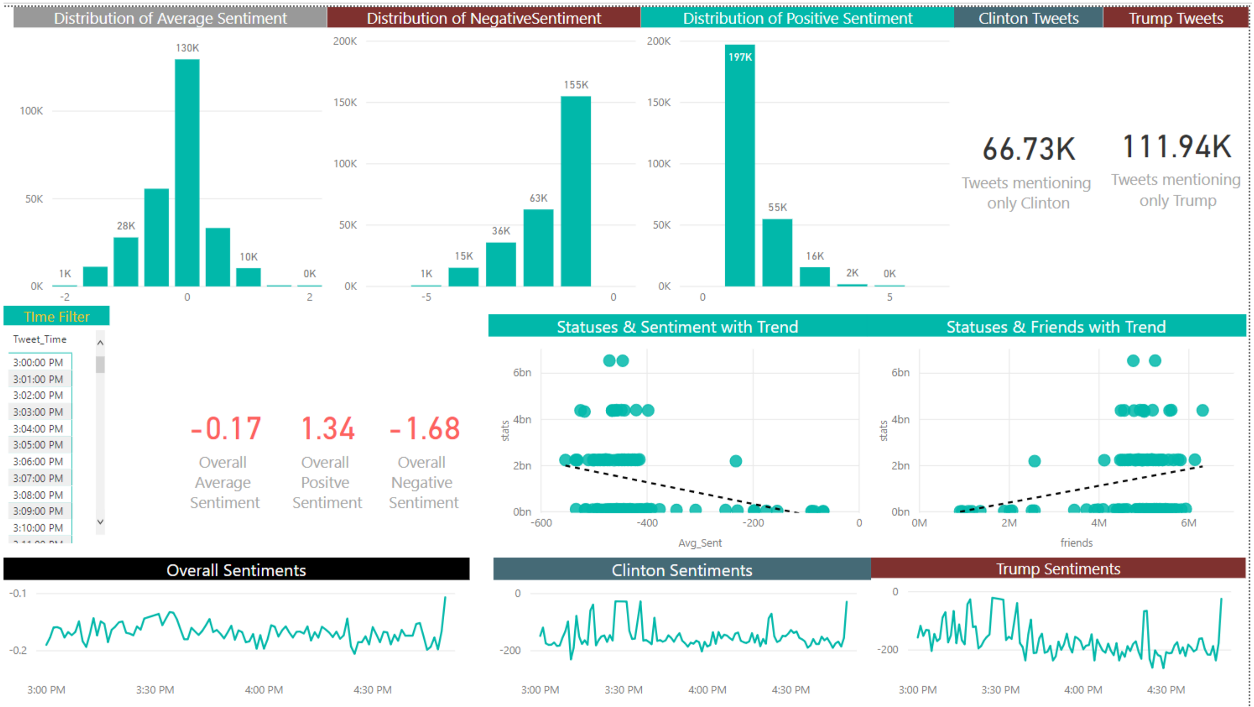


Figure 11 The Power BI dashboard showing AVG_SENT, NEG_SENT and POS_SENT distributions. Also shown are the number of total tweets for Clinton and Trump, the trend of Sentiments with the number of Statuses and a dynamic, selectable time-column on the far left

The Dashboard features *AVG_SENT, NEG_SENT and POS_SENT distributions. Also shown are the number of total tweets for Clinton and Trump, the trend of Sentiments with the number of Statuses and a dynamic, selectable time-column on the far left to manipulate graphs and provide information on the selected time period.*