Final Project:

Mueller Report Sentiment Analysis and Exploration

Sam Blumer

For my final project, I will be reviewing the Mueller report that was released a couple of weeks back. The analysis will be focused on exploring common words, ngrams, bigrams, and a sentiment analysis. For the sake of processing, the entire report was not analyzed. I attempted to download the report as a text file, but for purposes of the final results, I used the dataset and started code from here: https://trial.dominodatalab.com/u/johnjoo/Mueller-Report/browsem (https://trial.dominodatalab.com/u/johnjoo/Mueller-Report/browsem)

First Step will be to load the necessary packages and data.

Load Packages

In [84]:
```
import numpy as np
import pandas as pd
from os import path
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt
```

Import Data

In [2]:
```
data = pd.read_csv("mueller_report.csv", index_col=0)
```

In [3]:
```
data.head()
```

Out[3]:

page	line	text
1	1	U.S. Departme...
1	2	AttarAe:,c\\'erlc Predtiet // Mtt; CeA1:ttiA
1	3	Ma1...
1	4	Report On The Investigation Into
1	5	Russian InterferenceIn The

Data Exploration and Cleaning

Total Number of Words:

In [4]:
```
data['char_count'] = data['text'].str.len()
```

In [5]:
```
data.head()
```

Out[5]:

page	line	text	char_count
1	1	U.S. Departme...	59.0
1	2	AttarAe:,c\\'erlc Predtiet // Mtt; CeA1:ttiA	44.0
1	3	Ma1...	90.0
1	4	Report On The Investigation Into	36.0
1	5	Russian InterferenceIn The	39.0

Remove Empty Rows

In [6]:
```
data = data[data['text'].notnull()]
```

Lowercase

In [7]:
```
data['text'] = data['text'].apply(lambda x: " ".join(x.lower() for x in x.spli
t()))
data['text'].head()
```

Out[7]:
```
page
1                          u.s. department of justice
1       attarae:,c\\'erlc predtiet // mtt; cea1:ttia
1    ma1:ertalprn1:eeteduader fed. r. crhtt. p. 6(e)
1                   report on the investigation into
1                         russian interferencein the
Name: text, dtype: object
```

Remove Punctuation

In [8]:
```
data['text'] = data['text'].str.replace('[^\w\s]','')
data['text'].head()
```

Out[8]:
```
page
1                    us department of justice
1         attaraecerlc predtiet  mtt cea1ttia
1    ma1ertalprn1eeteduader fed r crhtt p 6e
1           report on the investigation into
1                 russian interferencein the
Name: text, dtype: object
```

Remove Stopwords

In [9]:
```
from nltk.corpus import stopwords
stop = stopwords.words('english')
data['text'] = data['text'].apply(lambda x: " ".join(x for x in x.split() if x
 not in stop))
data['text'].head()
```

Out[9]:
```
page
1                      us department justice
1         attaraecerlc predtiet mtt cea1ttia
1    ma1ertalprn1eeteduader fed r crhtt p 6e
1                       report investigation
1                     russian interferencein
Name: text, dtype: object
```

Common Words

In [10]:
```
freq = pd.Series(' '.join(data['text']).split()).value_counts()[:10]
freq
```

Out[10]:
```
president        1871
302              1666
trump            1432
us                909
2016              825
campaign          822
cohen             761
russian           757
justice           667
investigation     630
dtype: int64
```

Create Wordcloud

In [12]:
```
text = " ".join(str(line) for line in data['text']) # join text from rows into
 one string
```

In [17]:
```
stopwords = set(STOPWORDS) # set stopwords list

#create and plot wordsclouds
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(
text)

fig=plt.figure(figsize=(18, 16), dpi= 80, facecolor='w', edgecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

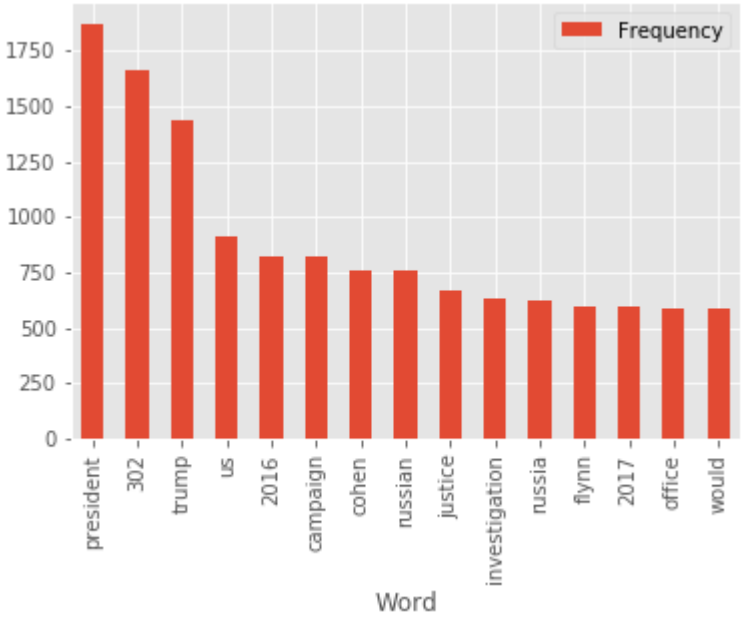
In addition to our wordcloud, we can create a graph and list of the top 15 most commonly used words in the report:

In [19]:
```
import nltk
import matplotlib
import string
nltk.download('punkt')
nltk.download('stopwords')
words = nltk.tokenize.word_tokenize(text)
words = [w.lower() for w in words]
word_dist = nltk.FreqDist(words)

stopwords = nltk.corpus.stopwords.words('english') + list(string.punctuation)
+ ['...', '``',"'s", "''"]
words_except_stop_dist = nltk.FreqDist(w for w in words if w not in stopwords)


print('All frequencies, NOT including STOPWORDS:')
print('=' * 60)
top_N = 15
rslt = pd.DataFrame(words_except_stop_dist.most_common(top_N),
                    columns=['Word', 'Frequency'])
print(rslt)
print('=' * 60)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\blume\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\blume\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

All frequencies, NOT including STOPWORDS:
============================================================
            Word  Frequency
0       president       1871
1             302       1666
2           trump       1432
3              us        909
4            2016        825
5        campaign        822
6           cohen        761
7         russian        757
8         justice        667
9   investigation        630
10         russia        620
11          flynn        600
12           2017        595
13         office        587
14          would        585
============================================================
```

In [20]:
```
rslt = pd.DataFrame(words_except_stop_dist.most_common(top_N),
                    columns=['Word', 'Frequency']).set_index('Word')

matplotlib.style.use('ggplot')

rslt.plot.bar(rot=90)
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x18c744f36d8>
```



Bigrams

To get another look at some of the term frequencies, we can create a dictionary of bigrams and their counts, then plot them in a wordcloud.

In [85]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

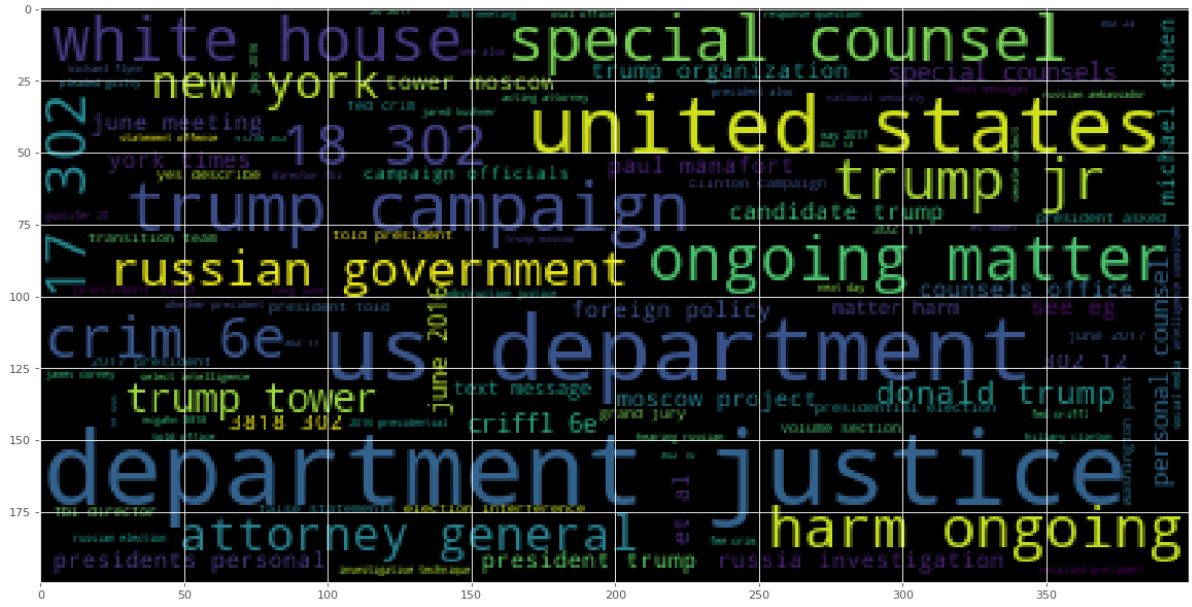
In [98]:
```
bigrams = CountVectorizer(ngram_range= (2,2))
counts = bigrams.fit_transform([text])
#(bigrams.vocabulary_)
```

In [99]:
```
story_counts = np.array(counts.todense()).flatten()
```

In [89]:
```
freq_dictionary = {}
for v, i in bigrams.vocabulary_.items():
    freq_dictionary[v] = story_counts[i]
```

In [102]:
```
wc = WordCloud()
fig=plt.figure(figsize=(18, 16), dpi= 80, facecolor='w', edgecolor='k')
plt.imshow(wc.generate_from_frequencies(freq_dictionary))
```

Out[102]: <matplotlib.figure.AxesImage at 0x18c01562c18>



Senitment Analysis

Because our data is structed in rows, we can iterate over the text in each row and provide a sentiment score:

In [49]:
```
data['sentiment'] = data['text'].apply(lambda text: TextBlob(text).sentiment)
```

If we look at the first 10 rows of our dataset, we will find that only 1 of them received a score:

In [51]:
```
data[0:10]
```

Out[51]:

page	line	text	char_count	sentiment
1	1	us department justice	59.0	(0.0, 0.0)
1	2	attaraecerlc predtiet mtt cea1ttia	44.0	(0.0, 0.0)
1	3	ma1ertalprn1eeteduader fed r crhtt p 6e	90.0	(0.0, 0.0)
1	4	report investigation	36.0	(0.0, 0.0)
1	5	russian interferencein	39.0	(0.0, 0.0)
1	6	2016 presidentialelection	39.0	(0.0, 0.0)
1	7	volume ii	50.0	(0.0, 0.0)
1	8	special counsel robert mueller iii	58.0	(0.35714285714285715, 0.5714285714285714)
1	9	submitted pursuant 28 cfr 6008c	66.0	(0.0, 0.0)
1	10	washington dc	53.0	(0.0, 0.0)

To find the most polarizing terms of the report, we can remove all rows that do not have a sentiment score, then order by highest polarity.

Each word in the lexicon has scores for:

1) polarity: negative vs. positive (-1.0 => +1.0)

2) subjectivity: objective vs. subjective (+0.0 => +1.0)

In [58]:
```
sentiments = data[data.sentiment != (0.0, 0.0)]
```

In [59]:
```
sentiments[0:10]
```

Out[59]:

page	line	text	char_count	sentiment
1	8	special counsel robert mueller iii	58.0	(0.35714285714285715, 0.5714285714285714)
3	6	special counsels investigation 11	151.0	(0.35714285714285715, 0.5714285714285714)
3	7	ii russian active measures social media campai...	152.0	(-0.03333333333333333, 0.2222222222222222)
3	11	1 ira ramps us operations early 2014 19	110.0	(0.1, 0.3)
3	12	2 us operations iracontrolled social media acc...	103.0	(0.03333333333333333, 0.06666666666666667)
3	17	5 us operations involving political rallies 29	126.0	(0.0, 0.1)
3	22	ill russian hacking dumping operations 36	152.0	(-0.25, 0.5)
3	34	b wikileakss first contact guccifer 20 dcleaks 45	107.0	(0.25, 0.3333333333333333)
4	16	2 potential campaign interest russian hacked m...	104.0	(0.0, 0.5)
4	36	c march 31 foreign policy team meeting 85	113.0	(-0.125, 0.125)

In [77]:
```
sentiments.sort_values('sentiment', ascending=False, inplace=True)
```

```
C:\Users\blume\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

In [79]:
```
sentiments[0:15]
```

Out[79]:

page	line	text	char_count	sentiment
261	24	tell awesome job wanted see	79.0	(1.0, 1.0)
92	7	excellent guy 421	25.0	(1.0, 1.0)
358	26	wonderful family michael businessman accountla...	95.0	(1.0, 1.0)
234	4	suffered one greatest defeats history politics 60	68.0	(1.0, 1.0)
445	17	mandate indictments ofmanafort gatess plea det...	100.0	(1.0, 0.3)
435	21	best knowledge mr trump never contact anyone p...	94.0	(1.0, 0.3)
147	12	email tuesday august 2 best tues weds nyc 9 17	66.0	(1.0, 0.3)
312	11	hicks raffel advised best strategy proactively...	103.0	(1.0, 0.3)
96	19	minister russia conveyed ivanov advice best ar...	100.0	(1.0, 0.3)
423	11	best knowledge mr trump never contact anyone	93.0	(1.0, 0.3)
121	23	best	6.0	(1.0, 0.3)
98	12	email wanted pass info along decide whats best	98.0	(1.0, 0.3)
148	22	prime minister plan emphasized yanukovych woul...	102.0	(0.9, 1.0)
109	8	stated would send readout soon regarding incre...	98.0	(0.9, 0.9)
378	38	incredible id 617 lower courts also rejected v...	89.0	(0.9, 0.9)

Interestingly, our most positive tweets (where the polarity is a solid 1.0) also seem to be the most subjective as well (also receiving a solid 1.0).

Overall Sentiment of Report

In [48]:
```
from textblob import TextBlob
testimonial = TextBlob(text)
testimonial.sentiment

#testimonial.sentiment.polarity
```

Out[48]:
```
Sentiment(polarity=0.03890388355605746, subjectivity=0.36562662974954235)
```

Conclusion

Overall, the entire report recieved a polarity score of .03, which is just about 0, meaning that the report itself is not necessarily positive nor is it negative. Additionally, the full report recieved a subjectivity score which could be, on a scale from 0-1, with 0 being objective, of .36, which would be closer to objective than subjective. I was actually surprised by these numbers, despite the fact that the report is intended to be written by a neutral party. As we might have expected, given the subject of the investigation, some of the words that occur most commonly are "Trump", "Russia", "Justice", and "Cohen". Even with regard to the bigrams, we see similar patterns, with common phrases being "russian government", "special counsel", and "white house". Interestingly, we see that "ongoing matter" is also a common bigram, which is the term that was placed over all redcations in the report, which gives evidence to just how much of the report was redacted.

In [ ]: