**Final Project – A sentiment analysis of CEO twitter data**

Daniel Pearl

2020-06-02

**Project Description**

The goal of this project is to gather tweets from various CEOs, save them to a local MongoDB database, and run a sentiment analysis for each one. A word cloud will also be generated from individual CEO tweets. Initially, I had intended to do this with word leader tweets, but it became difficult to find the handles I required. Instead, I found a website that listed all the popular CEO twitter handles. The final output will be a dictionary of all the CEO names and their sentiment scores. These scores will be the output from tweets broken down into word tokens of a particular day.

**Sentiment Analysis**

The textblob sentiment analysis will return two values, Polarity and Subjectivity. Polarity has to do with the tone of the text or whether the text is positive, negative, or neutral. These values are determined based on the number of weighted emotional words included in the document. The subjectivity is how subjective vs objective the document might be. This might include the personal dogma or feelings of the writers. For the purposes of this project, I will be using the polarity score.

**Questions:**

Because of this updated application for the project, I decided to limit the number of questions:

* What are the sentiment scores for each of the CEOs on a given day?
* Which CEO has the highest sentiment score? Which has the lowest?
* What does a positive word cloud look like compared to a negative one?

**Data Sources**

The data sources for this project will be the twitter API for the CEO tweets and the list of twitter handles found on ceoworld.biz:

[https://ceoworld.biz/2018/01/02/top-ceos-and-business-leaders-on-twitter-you-should-be-following/.](https://ceoworld.biz/2018/01/02/top-ceos-and-business-leaders-on-twitter-you-should-be-following/)

**Preprocessing**

Invalid twitter handles are removed from the CEO URL. I replaced some of the deleted handles with valid ones by appending them back into the list. The tweets were not altered before saving them to the MongoDB database. Once extracted from the database, the CEO data is converted to a dictionary for each processing step.

**Methods**

The first part of this program involves creating a web crawler using the package BeautifulSoup. To do this, I created a function called get\_ceo\_handles:

# This function retrieves the top CEO twitter handles from ceoworld.biz

def get\_ceo\_handles():

# Retrieve URL

url = "https://ceoworld.biz/2018/01/02/top-ceos-and-business-leaders-on-twitter-you-should-be-following/"

# Get HTTP response

response = request.urlopen(url)

# Convert response to string

html = response.read().decode('utf-8')

# Parse string

soup = BeautifulSoup(html, "html.parser")

# Append CEO handles to a list

ceo\_handles = []

tr = (soup.findAll('tr'))

for td in tr:

tag = td.findAll('td')[1].string

if tag and tag[0] == "@":

ceo\_handles.append(tag)

# Return twitter handles

return(ceo\_handles)

Next, after these CEOs were saved to a variable, I removed the invalid handles and replaced them with the valid ones. Because the link was from 2018, some of the information was no longer valid. I created a function called get\_tweets that takes in the list of CEO twitter handles and returns a list of tweets. This function uses the consumer and oauth keys to call the twitter api using tweepy.

# This function extracts data from twitter

def get\_tweets(user\_list):

# Get authorization information

CONSUMER\_KEY = "XXXX"

CONSUMER\_SECRET = "XXXX"

OAUTH\_TOKEN = 'XXXX

OAUTH\_SECRET = "XXXX"

# Authorization using consumer key and consumer secret

auth = tweepy.OAuthHandler(CONSUMER\_KEY, CONSUMER\_SECRET)

# Access to user's oauth key and oauth secret

auth.set\_access\_token(OAUTH\_TOKEN, OAUTH\_SECRET)

# Calling api

api = tweepy.API(auth, wait\_on\_rate\_limit=True)

# 50 tweets to be extracted

tweet\_list = []

for username in user\_list:

tweets = (api.user\_timeline(screen\_name=username, count = 50, include\_rts = False, tweet\_mode = 'extended'))

for tweet in tweets:

# Appending tweets to the empty array tmp

tweet\_list.append(tweet.\_json)

# Printing the tweets

return(tweet\_list)

Once the tweets were extracted and saved as a list, they were saved to a MongoDB database called ‘tweets’ with a collection called ‘ceos’. I used the ‘save\_to\_DB’ function supplied in class for homework 2 to do this. To extract the tweets from the database, I created a function called get\_tweets\_DB that takes in the date the tweets were created:

# Retrieves tweets from mongoDB database

def get\_tweets\_DB(dt):

# Get local mongo client

client = pymongo.MongoClient('localhost', 27017)

# Get database

db = client.tweets

# Get collection

coll = db.ceos

# Get tweets

tweets = coll.find()

# Appends tweets to a list from the date

tweet\_list = []

for tweet in tweets:

if tweet['created\_at'][:10] == dt:

tweet\_list.append(tweet)

# Returns list of tweets

return tweet\_list

Once these tweets are extracted from the database, I created a dictionary with the CEO names as the keys and their tweets as the key values. In this format, it is easier to tokenize the text for each of the CEOs. I created a tokenize function that takes in the dictionary of CEOs and returns a list of unique words for the CEO of interest. This function uses the TweetTokenizer from the nltk package to parse each of the words from the tweets. Stop words and punctuation are omitted from the filtered tokens.

# Tokenizes words for CEO

def tokenize(ceos, ceo):

# Separate all words into list

tk = TweetTokenizer()

token = tk.tokenize(ceos[ceo])

# Convert all tokens to lowercase

token = [token.lower() for token in token]

# Get stop words

stop\_words = set(stopwords.words('english'))

punctuation = string.punctuation

# Remove stopwords, punctuation, and non-english words

filtered\_token = []

for word in token:

if word not in stop\_words:

if word not in string.punctuation:

if 'https://' not in word:

filtered\_token.append(word)

# Return list of filtered words

return filtered\_token

This now tokenized list is used to create a word cloud from the wordcloud package. The WordCloud function from this package takes in the following parameters:

wordcloud = WordCloud(

background\_color='white',

stopwords=set(stopwords.words('english')),

max\_words=100,

max\_font\_size=40,

scale=3

).generate(str(tk))

Using the tokens from Elon Musk, the following word cloud is produced:

*A screen shot of a social media post

Description automatically generated*

Finally, in order to get the sentiment score for each CEO, the TextBlob function from the textblob package was used. This function takes in the dictionary of CEO names with their tokens and returns a dictionary of CEO names with their polarity scores. The following output is produced from tweets on Mon, May 25, 2020:

{'Elon Musk': -0.1889,

'Tim Cook': 0.4,

'Lord Sugar': -0.1208,

'John Legere': -0.0404,

'Aaron Levie': 0.125,

'Marc Benioff': 0.1426,

'Bryan Kramer - Keynote Speaker': 0.4625,

'Ashley Alexiss': 0.5,

'MarceloClaure': 0.2762,

'John Hall': 0.4,

'Flint Bedrock 🇿🇼🇬🇧': 0.9,

'John Lincoln': 0.375,

'Gurbaksh Singh Chahal': 0.125,

'Mike Kawula': 0.2942,

'@shellykramer': 0.0,

'Michael Brenner': 0.2805,

'Nika Stewart': 0.0,

'Danielle Morrill': 0.2796,

'Josef Holm': 0.0,

'Keith Krach': 0.3187,

'Peter Bordes': 0.3333,

'Rob Peters': 0.2333,

'Dr. Ganapathi Pulipaka': 0.0867,

'David R. Prasser': 0.2874,

'MariAnne Vanella': 0.0966,

'david jones': -0.225,

'Chuck Robbins': 0.3982,

'Tim Jackson CAE': 0.2161,

'Mauro Biasolo': 0.0,

'Jon Ferrara': 0.2425,

'Gil Eyal': 0.0,

'Stephen Kelly': 0.38,

'Helena Morrissey DBE': 0.3355,

'Larry Kim': 0.0

**Wordclouds**

For the negative word cloud, I chose to represent Lord Sugar, a British businessman and media personality. I looked up his recent tweets and found that he had been criticizing another tv personality, Piers Morgan. This criticism negatively impacted his polarity score. Polarity score: -0.1208

A picture containing table, wooden, laptop, green

Description automatically generated

For the positive word cloud, I chose to represent Bryan Kramer, a Papua New Guinea politician and businessman. Words like leadership, kind, and best are likely to result in a higher polarity score. Polarity score: 0.4625

A picture containing bottle, table, holding, people

Description automatically generated

**Conclusion**:

For May 25, 2020, of the CEOs listed, Flint Bedrock has the highest polarity score of 0.9 while David Jones has a polarity score of -0.225. For the majority of CEOs listed, the polarity score is positive. This makes sense because the information the CEOs tweet can directly impact the sales of their products/services. There are a few CEOs with negative polarity scores: Elon Musk, Lord Sugar, Tim Cook, and John Kegere. One thing that might affect the sentiment analysis is the amount of words or tokens listed for each CEO. CEOs with fewer tweets and fewer words per tweet might be offset by just a few words. The confidence of the sentiment analysis is increased with more words to analyze. Perhaps by analyzing a larger number of tweets on multiple days, a more accurate analysis can be obtained. I have chosen to create word clouds from CEOs who are on opposite sides of the polarity score and who have a sufficient number of tweets/tweet lengths to ensure some confidence.

One potential application for the CEO sentiment analysis would be determining whether or not to buy stock in a specific company. Though it is out of scope for this project, it would be interesting to determine if the sentiment score was correlated with fluctuations in the stock market. A regression could be created in order to determine what amount of change in the stock market is attributed to the sentiment score of the CEO’s tweets. If that correlation is significant, an algorithm can be created to buy and sell stock depending on a sentiment score threshold.