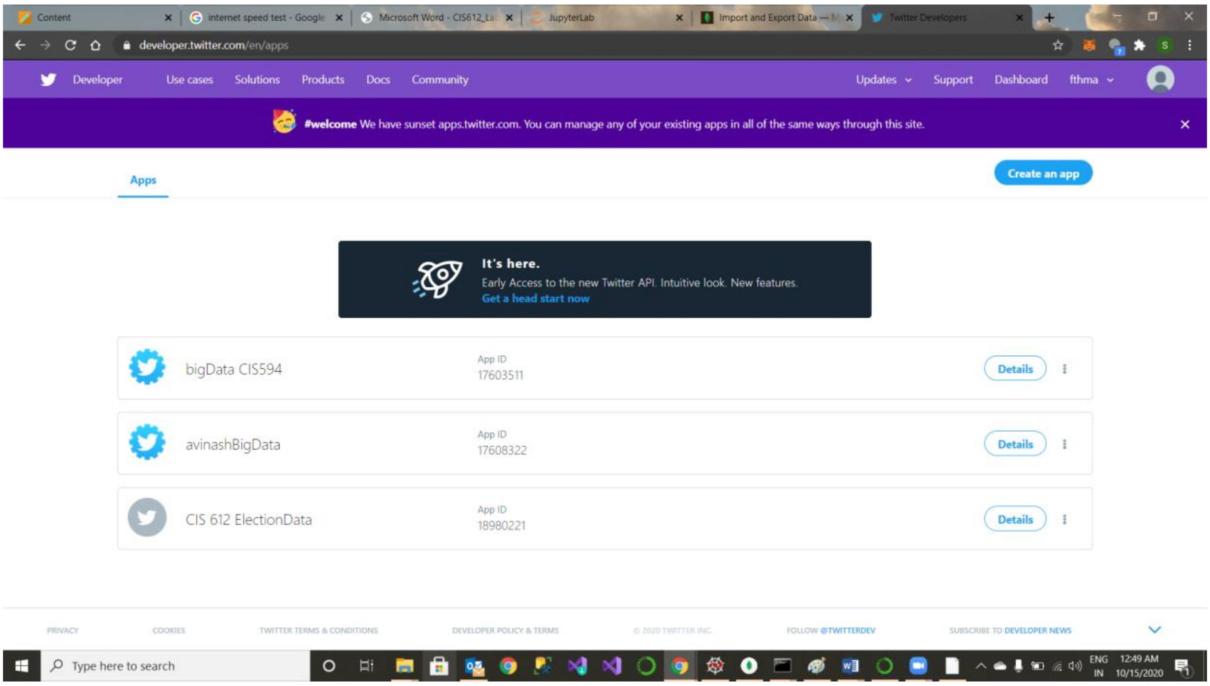
Sentiment Analysis on Twitter 2020 election data

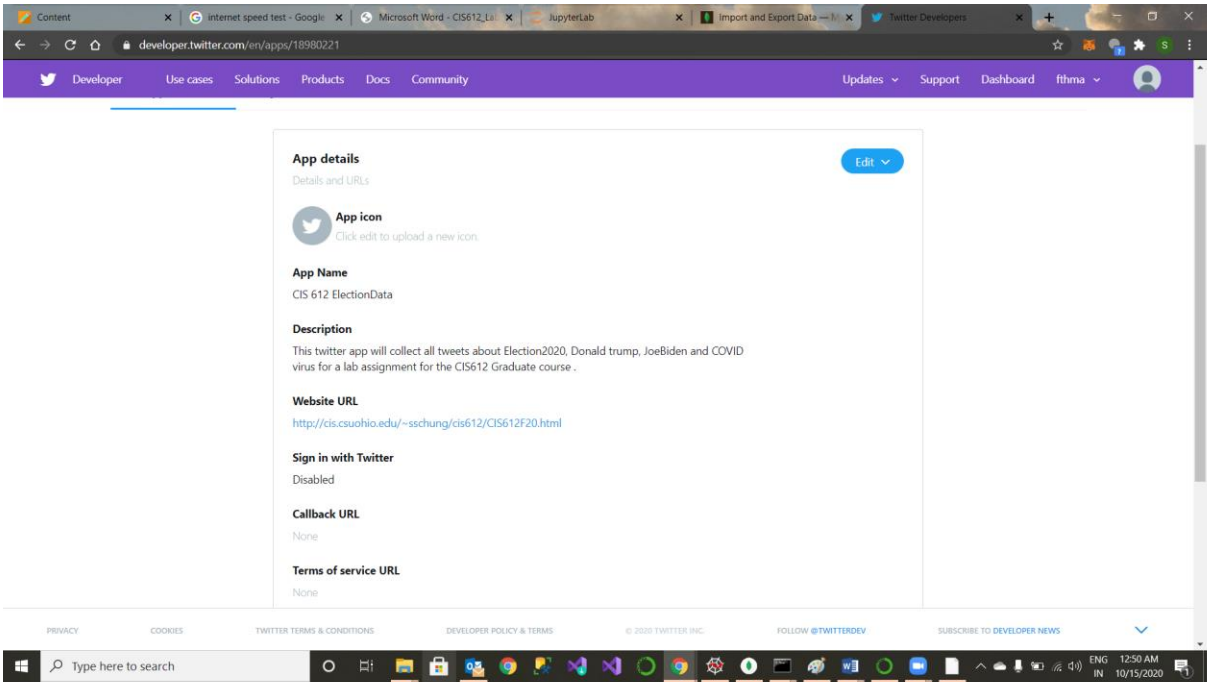
Fathima Syeda

Platform Setup:

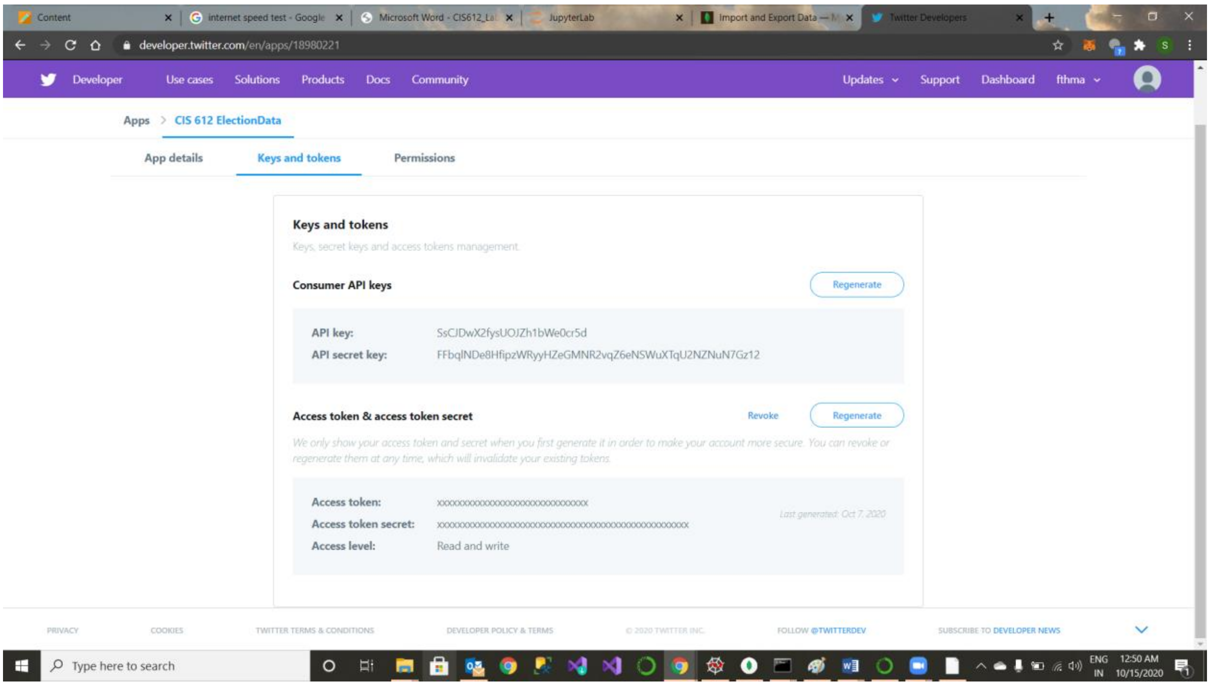
We must first set up a twitter developer account.



Create an app in you twitter account to fetch the data for your lab.

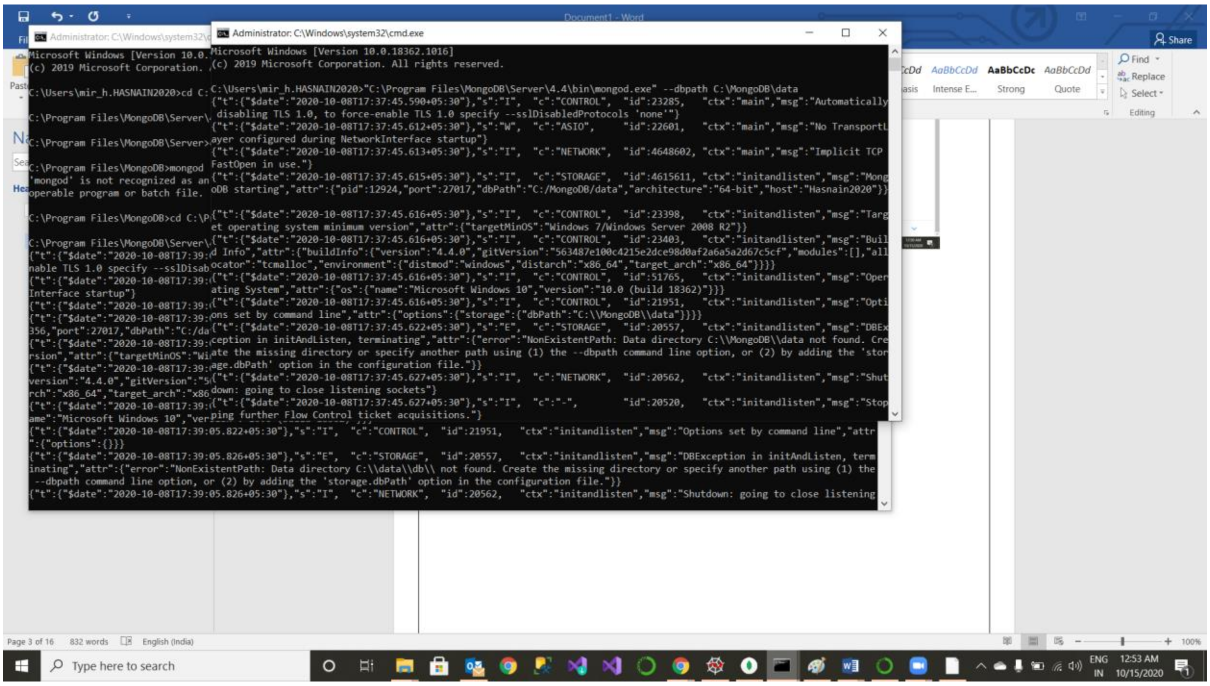


Get the Consumer API keys and generate the Access tokens to use in the python script to get tweets

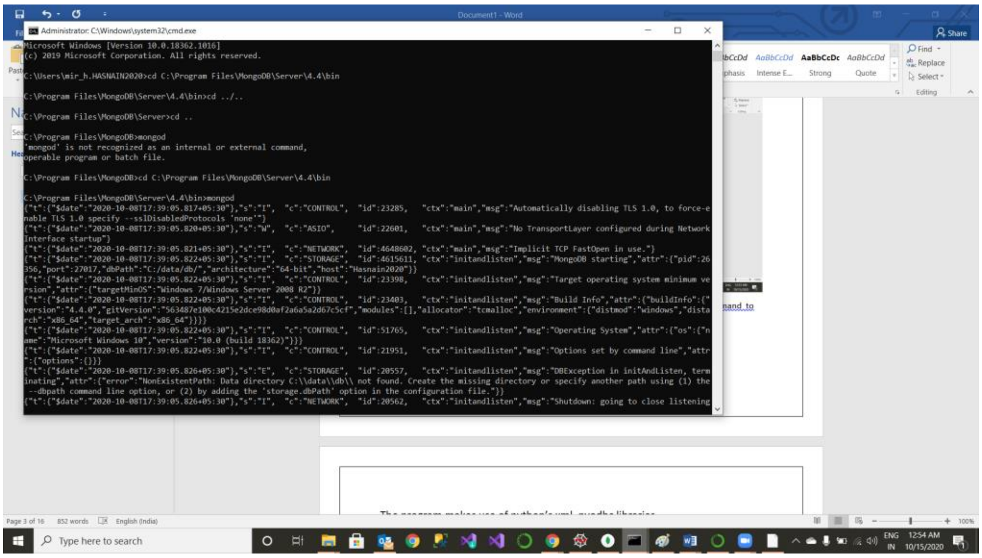


Also set up mongodb to store the tweets into it.

To turn on the server go to the bin folder of the mongodb server in your system and ryoe the following command **mongod.exe" --dbpath C:\MongoDB\data**



Now open a new window and browse to the mongodb folder again and typr **mongod** command to start the server.



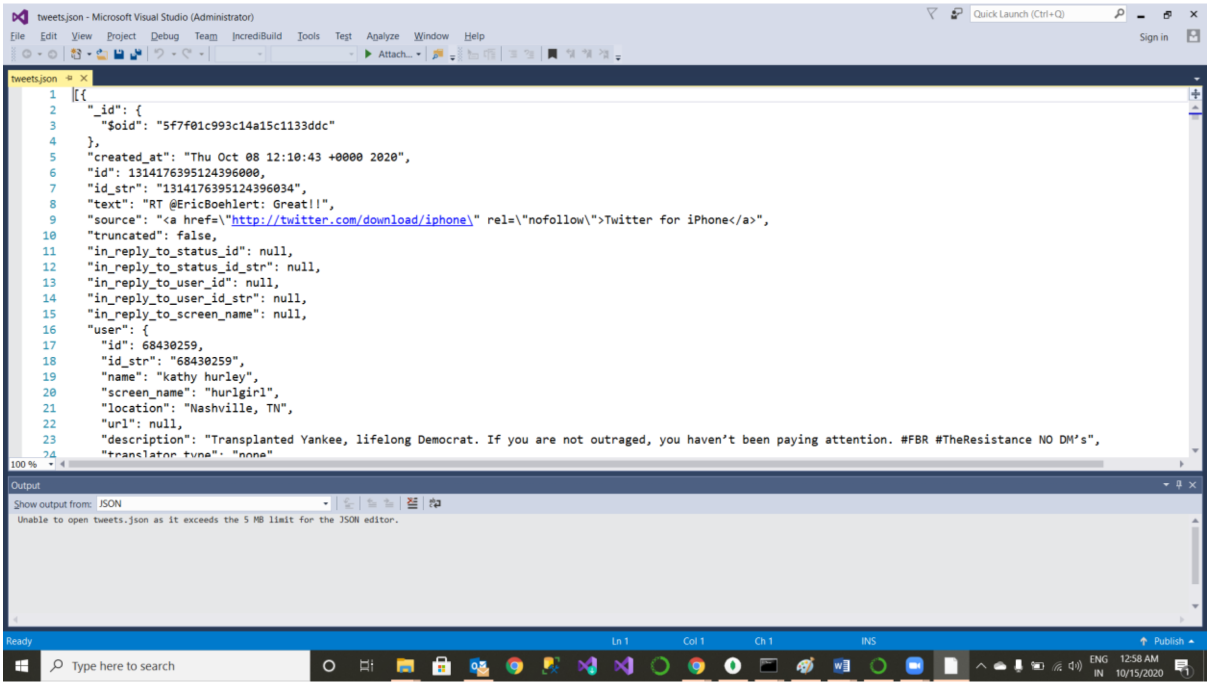
Data Collection:

Our twitter data was collected on two hastags, #joebiden and #donaldtrump. The dataset consists of around 900,000 tweets on joe biden and around 700,000 tweets in trump.

We use python’s Tweepy library to connect to the twitter api and stream live tweets.

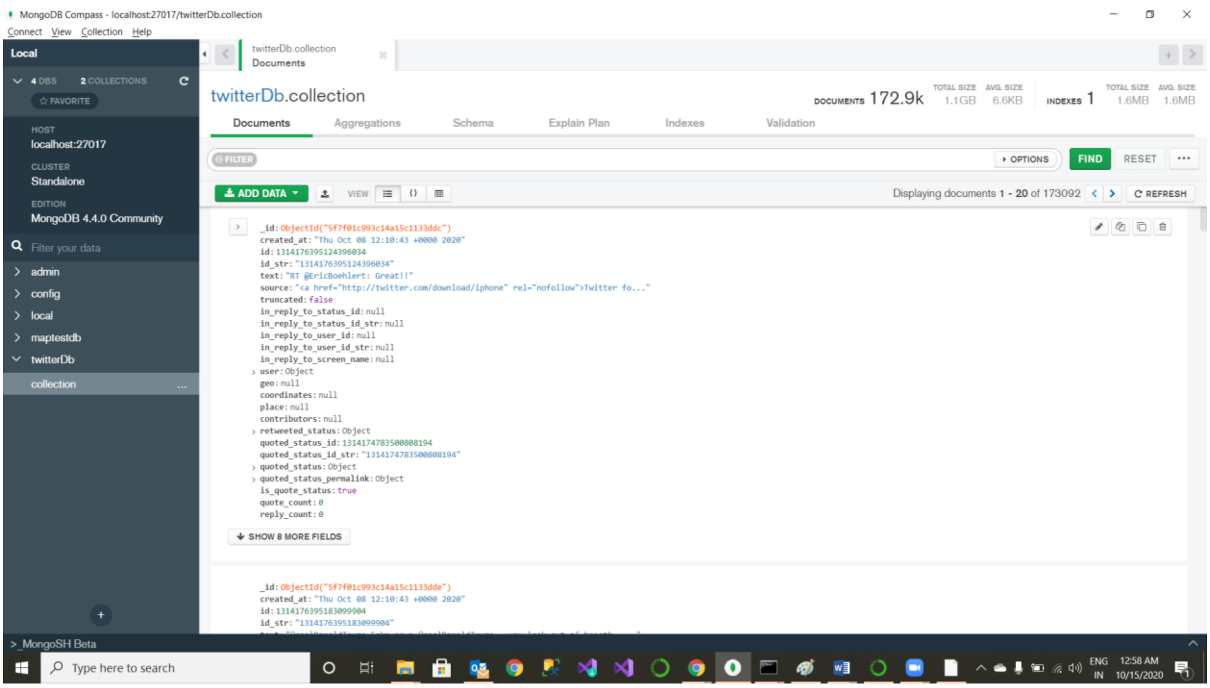
We filter out the tweets using the following keywords:

filter(track=["#DonaldTrump","#JoeBiden"],languages=["en"])



The python script is run on different days for around an hour each day , to collect the tweets.

These tweets are directly inserted into Mongodb .



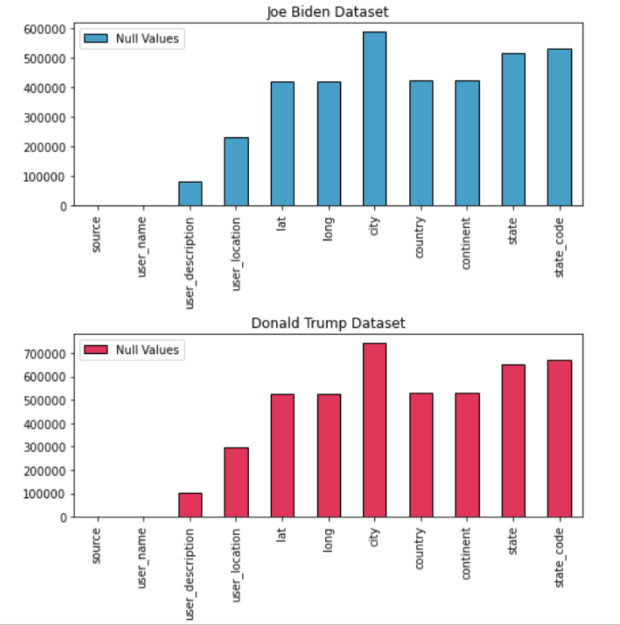
Once enough tweets had been collected, we divided the tweets based on the candidates into two csv files.

Exploring the data and Pre-processing:

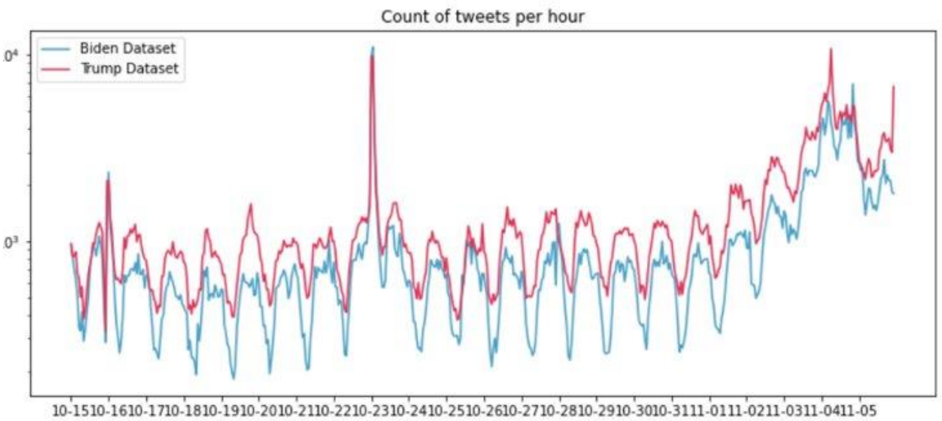
The tweets have 21 fields but not all of the tweets are relevant to our sentiment analysis and hence we shall only be using a subset of it.

- We take the user_join_date and created_at fields and convert them into pandas date time.
- Then we normalise the number of likes and retweets to allow fair analysis by dividing the number of likes and tweets by the difference in collected_at and created_at.

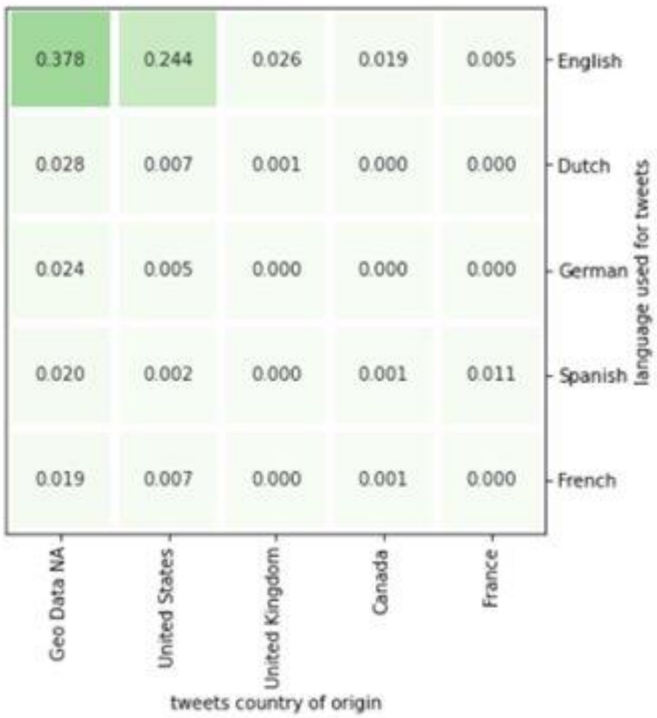
Below is a visualisation for the null value tweets in our dataset for joebiden and donald trump.



We also visualize the count of tweets per hour below. We notice that most tweets with geodata info are from America . Tweets about both candidates were stable in terms of tweets per hour and pattern regularity, except nearing the dates of election day. On the election date itself there is large notable changes in the pattern regularity on both datasets, with an steady increasing trend in tweet volume for the both of the presidential candidates.



We also visualise in the form of a heatmap the number of tweets per continent of origin and the language tweeted in. We used the ***Langdetect*** function to sample 4000 tweets to find the most common languages of the tweet. The heatmap above only illustrates the top 5 languages used and the top 5 countries that they were tweeted from, in all 40 languages were detected with English accounting for almost 80% of the tweets.



We get the retweet_count ,liked from extended Tweets json object and the Lat, long from the geo_location dataset from USA.

We drop those tweets that have null values for our required fields and clean the tweets to r**emove stopwords, strings with "http" e**tc and then **lemmatize** the words.

The cleaned final dataset which is ready for sentiment analysis looks as follows with the following 7 fields.

	tweet_id	likes	retweet_count	user_id	user_followers_count	lat	long
count	9.709190e+05	970919.000000	970919.000000	9.709190e+05	9.709190e+05	445719.000000	445719.000000
mean	1.322494e+18	7.477011	1.698500	4.468311e+17	2.260357e+04	35.697936	-40.369638
std	2.555133e+15	158.058117	40.028419	5.544702e+17	3.042152e+05	18.823129	67.531751
min	1.316529e+18	0.000000	0.000000	5.310000e+02	0.000000e+00	-90.000000	-175.202642
25%	1.320478e+18	0.000000	0.000000	2.169326e+08	7.700000e+01	32.701939	-96.796856
50%	1.323612e+18	0.000000	0.000000	2.373392e+09	4.410000e+02	39.783730	-74.006015
75%	1.324505e+18	1.000000	0.000000	1.078344e+18	2.066000e+03	46.603354	6.776314
max	1.325589e+18	74084.000000	20491.000000	1.325581e+18	1.911533e+07	90.000000	179.048837

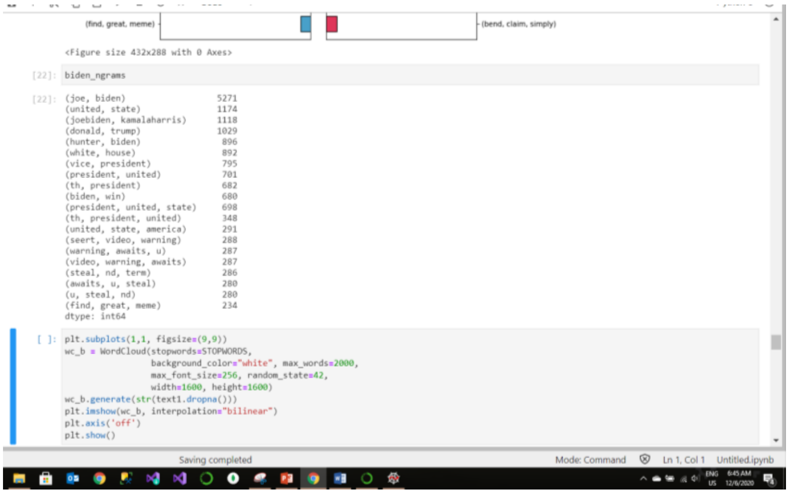
Sentiment Analysis (VADAR)

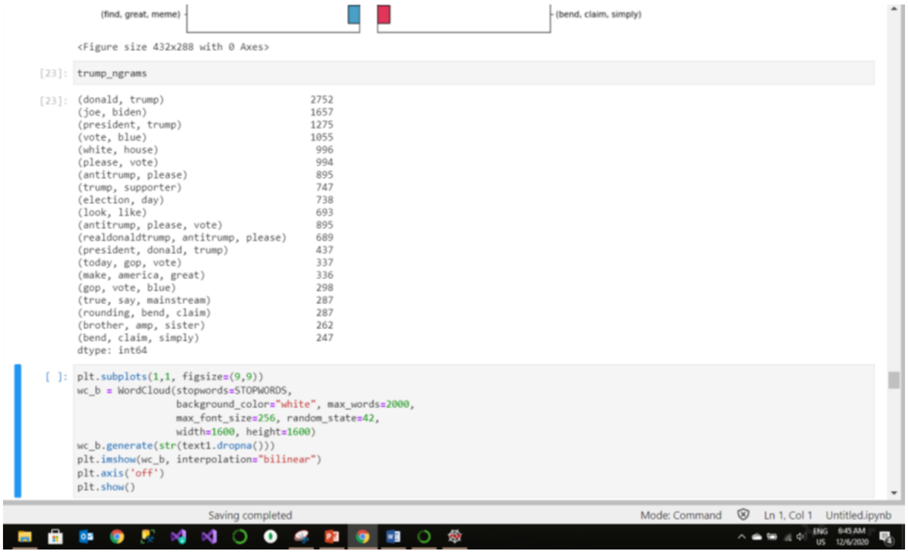
- VADER (Valence Aware Dictionary and sentiment Reasoner) package, which is a lexicon and rule-based sentiment analysis tool
- specifically tuned to sentiments expressed in social media
- sensitive to both polarity (positive/negative)
- relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores.
- Imported from nltk and can be applied directly to unlabeled text data.

Finding the Ngrams of the tweets:

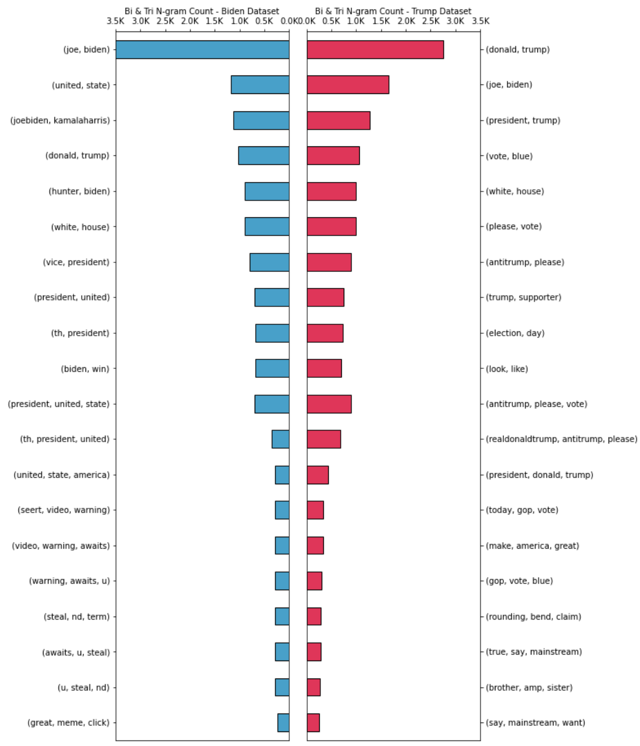
Once the dataset is clean and ready, we find the 2grams and 3grams for biden and trump tweets.

We find the top 10 bigrams and trigrams for the dataset for the both the candidates.





The following is a plot of top 10 ngrams .



Sentiment Analysis compound score:

The VADAR analysis produces a compound score so we took the mean compound score for the most recent 14 days and the first 14 days for each state.

The results seems to show a large number of states are trending to a “Positive” sentiment score for the democratic candidate from the previous more “Neutral” sentiment(sentiment score between 0.05 and -0.05). Whereas most states were still largely "Neutral" for the republican candidate.

The below visualization was generated by first assigning each tweet a "Positive", "Neutral" or "Negative" sentiment then summing those for each day and calculating the proportions for each sentiment group. Then using logistic regression to find the best fit line to better show the sentiment trend overtime. When viewing the results:

The trend over the entire timeframe of the dataset for both presidential candidates is an increasing "Positive" and "Neutral" sentiment, with reducing negative sentiment.

Near to the election day we see a "Positive" sentiment increase quicker for the democratic candidate over the republican where we see a noticeable gap develop in the logistic regression lines.

Moving onto the "Neutral" sentiment, the differences between two candidates remain largely steady until there is a noticeable blip on election day where the gap briefly disappears before then returning to the previous steady difference.

Post-election day we see a shape increase in "Negative" sentiment for the republican candidate

Conclusion

We used the us election 2020 dataset to perform sentiment analysis only on data that had geo-data originating from the "United States of America" to find the sentiment in tweets about each presidential candidate. When reviewing sentiment at the state level as we approached the election date a large number of states were trending to a “Positive” sentiment score for the democratic candidate from the previously more “Neutral” sentiment. Whereas most states are still largely “Neutral” for the republican candidate.

Source Code:

```
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import json
from pymongo import MongoClient
```

```
#consumer key, consumer secret, access token, access secret.
ckey="SsCJDwX2fysUOJZh1bWe0cr5d"
csecret="FFbqlNDe8HfipzWRyyHZeGMNR2vqZ6eNSWuXTqU2NZNuN7Gz12"
atoken="855746302952583168-YRmeZRXU3FkBk1Dw3RHMFaaho1EDKiv"
asecret="7ypJOY3VB0Li4dFB1ZyJP8O4oNfkWLN1ekgBsx2m46wVU"


class listener(StreamListener):

    def on_data(self, data):
        all_data = json.loads(data)

        client=MongoClient('localhost',27017)
        if(client):
            print("connected to mongodb")

        db=client.twitterDb
        collection=db.Tweets_Collection
        db.collection.insert_one(all_data)


        tweet = all_data["text"]

        username = all_data["user"]["screen_name"]

        #with open('tweets.json','a') as Tweetsfile:
        #    Tweetsfile.write()


        #print((username,tweet))
        print(tweet)



        return True
```

```
    def on_error(self, status):
        print (all_data)

auth = OAuthHandler(ckey, csecret)
auth.set_access_token(atoken, asecret)

twitterStream = Stream(auth, listener())
twitterStream.filter(track=["Donald Trump","POTUS Covid","us election 2020","Covid-19 USA","Joe biden"],languages=["en"])

import os
import time
import missingno as msno
import pandas as pd
#import geopandas as gpd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import nltk, re, string, collections, unicodedata


from matplotlib import cm, dates
from matplotlib.ticker import ScalarFormatter
from matplotlib.ticker import FuncFormatter
from datetime import datetime, timedelta
from textblob import TextBlob
from wordcloud import WordCloud, STOPWORDS
from langdetect import detect
from nltk.util import ngrams
from nltk.tokenize import word_tokenize
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import subjectivity
from nltk.sentiment import SentimentAnalyzer
from nltk.sentiment.util import *
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

```
tweets_biden = pd.read_csv('hashtag_joebiden.csv', lineterminator='\n',
parse_dates=True)
tweets_trump = pd.read_csv('hashtag_donaldtrump.csv', lineterminator='\n',
parse_dates=True)
# Clean data
tweets_biden['country'].replace({'United States':'United States of America'},
inplace=True)
tweets_trump['country'].replace({'United States':'United States of America'},
inplace=True)


# Add Features
def normalise(x,y):
    x = np.array(x)
    y = np.array(y)
    return np.where(x == 0, 0, x / y)

def sentiment(data):
    temp=[]
    for row in data:
        tmp=sid.polarity_scores(row)
        temp.append(tmp)
    return temp

# convert to datetime object
tweets_biden['user_join_date']=pd.to_datetime(tweets_biden['user_join_date
'])
tweets_trump['user_join_date']=pd.to_datetime(tweets_trump['user_join_dat
e'])
tweets_biden['collected_at']=pd.to_datetime(tweets_biden['collected_at'])
tweets_trump['collected_at']=pd.to_datetime(tweets_trump['collected_at'])
tweets_biden['created_at']=pd.to_datetime(tweets_biden['created_at'])
tweets_trump['created_at']=pd.to_datetime(tweets_trump['created_at'])

# create additional date time columns
tweets_biden['created_at_r']=tweets_biden['created_at'].dt.strftime('%Y-%m-
%d %H')
```

```
tweets_trump['created_at_r']=tweets_trump['created_at'].dt.strftime('%Y-
%m-%d %H')
tweets_biden['created_at_r2']=tweets_biden['created_at'].dt.strftime('%m-
%d')
tweets_trump['created_at_r2']=tweets_trump['created_at'].dt.strftime('%m-
%d')

# normalise likes and retweets to allow fair analysis
b_tdiff=(tweets_biden['collected_at'] - tweets_biden['created_at'])
t_tdiff=(tweets_trump['collected_at'] - tweets_trump['created_at'])
b_tdiff=(b_tdiff.dt.days * 24 + b_tdiff.dt.seconds / 3600)
t_tdiff=(t_tdiff.dt.days * 24 + t_tdiff.dt.seconds / 3600)

# Use numpy vectorisation to create new columns for normalised likes and
retweets
tweets_biden['likes_norm'] = normalise(tweets_biden['likes'],b_tdiff)
tweets_biden['retweet_norm'] =
normalise(tweets_biden['retweet_count'],b_tdiff)
tweets_trump['likes_norm'] = normalise(tweets_trump['likes'],t_tdiff)
tweets_trump['retweet_norm'] =
normalise(tweets_trump['retweet_count'],t_tdiff)

# Visualisation args
cmap = sns.diverging_palette(0, 230, 90, 60, as_cmap=True)
barcolors =
['#87B88C','#9ED2A1','#E7E8CB','#48A0C9','#2A58A1','#2E8B55','#DF3659','Gre
y']
barstyle = {"edgecolor":"black", "linewidth":1}
heatmap1_args = dict(annot=True, fmt='.0f', square=False,
cmap=cm.get_cmap("RdGy", 10), center = 90, vmin=0, vmax=10000, lw=4,
cbar=False)
heatmap2_args = dict(annot=True, fmt='.3f', square=False, cmap="Greens",
center = 0.5, lw=4, cbar=False)
heatmap3_args = dict(annot=True, fmt='.0f', square=False, cmap=cmap, center
= 9200, lw=4, cbar=False)

def hide_axes(this_ax):
   this_ax.set_frame_on(False)
```

```
    this_ax.set_xticks([])
    this_ax.set_yticks([])
    return this_ax

def draw_heatmap1(df,this_ax):
    hm = sns.heatmap(df, ax = this_ax, **heatmap1_args)
    this_ax.set_yticklabels(this_ax.get_yticklabels(), rotation=0)
    this_ax.yaxis.tick_right()
    this_ax.yaxis.set_label_position("right")
    for axis in ['top','bottom','left','right']:
        this_ax.spines[axis].set_visible(True)
        this_ax.spines[axis].set_color('black')
    return hm

def draw_heatmap2(df,this_ax):
    hm = sns.heatmap(df, ax = this_ax, **heatmap2_args)
    this_ax.set_yticklabels(this_ax.get_yticklabels(), rotation=0)
    this_ax.yaxis.tick_right()
    this_ax.yaxis.set_label_position("right")
    for axis in ['top','bottom','left','right']:
        this_ax.spines[axis].set_visible(True)
        this_ax.spines[axis].set_color('black')
    return hm
def draw_heatmap3(df,this_ax):
    hm = sns.heatmap(df, ax = this_ax, **heatmap3_args)
    this_ax.set_yticklabels(this_ax.get_yticklabels(), rotation=0)
    this_ax.yaxis.tick_right()
    this_ax.yaxis.set_label_position("right")
    for axis in ['top','bottom','left','right']:
        this_ax.spines[axis].set_visible(True)
        this_ax.spines[axis].set_color('black')
    return hm

def thousands1(x, pos):
    'The two args are the value and tick position'
    return '%1.0fK' % (x * 1e-3)

formatterK1 = FuncFormatter(thousands1)
```

```
def thousands2(x, pos):
   'The two args are the value and tick position'
   return '%1.1fK' % (x * 1e-3)

formatterK2 = FuncFormatter(thousands2)

na_vals_b=pd.DataFrame({'Null Values':tweets_biden.isna().sum()})
na_vals_b=na_vals_b.loc[na_vals_b['Null Values'] > 0]
na_vals_t=pd.DataFrame({'Null Values':tweets_trump.isna().sum()})
na_vals_t=na_vals_t.loc[na_vals_t['Null Values'] > 0]

# Null values visualisation for tweets about Joe Biden and Donald Trump
fig, ax=plt.subplots(2,1, figsize=(8,8), gridspec_kw={'hspace':0.7})

na_vals_b.plot.bar(color=barcolors[3], **barstyle, ax=ax[0])
ax[0].set_title('Joe Biden Dataset')
ax[0].set_xticklabels(ax[0].get_xticklabels(), rotation=90)

na_vals_t.plot.bar(color=barcolors[6], **barstyle, ax=ax[1])
ax[1].set_title('Donald Trump Dataset')
ax[1].set_xticklabels(ax[1].get_xticklabels(), rotation=90)
plt.show()

source_df=pd.concat([tweets_biden[['source','tweet','country']].copy(),tweets
_trump[['source','tweet','country']].copy()])
source_df['country'].replace({'United States of America':'United States'},
inplace=True)
source_df=source_df.fillna('Geo Data NA')
source_df=source_df.drop_duplicates()

sources=pd.DataFrame(source_df.groupby(['source'])['tweet'].count().sort_val
ues(ascending=False)[:6])
sources=sources.reset_index()
sourceslst=sources.source.to_list()

country=pd.DataFrame(source_df.groupby(['country'])['tweet'].count().sort_va
lues(ascending=False)[:6])
country=country.reset_index()
```

```
countrylst=country.country.to_list()

platXtab=pd.DataFrame(source_df.groupby(['source','country'])['tweet'].count(
).unstack().fillna(0))

fig, ax=plt.subplots(2,2, figsize=(9,9),
                  gridspec_kw={'height_ratios':[2,5], 'width_ratios':[2,5],
'wspace':0.1, 'hspace':0.1})

hide_ax = ax[0,0]
hide_axes(hide_ax)

hm_ax = ax[1,1]
draw_heatmap1(platXtab.loc[sourceslst,countrylst], hm_ax)
hm_ax.set_xlabel('tweets country of origin')
hm_ax.set_ylabel('platform used for tweets')
hm_ax.set_yticklabels(('Twitter
Web','iPhone','Android','iPad','TweetDeck','Hootsuite'), rotation=0)

bar_ax = ax[0,1]
platXtab.loc[sourceslst,countrylst].sum().plot.bar(ax=bar_ax,
color=barcolors[1],**barstyle)
bar_ax.set_xlabel(bar_ax.get_xlabel())
bar_ax.xaxis.tick_top()
bar_ax.xaxis.set_label_position("top")
bar_ax.yaxis.set_major_formatter(formatterK1)
bar_ax.set_xticklabels(('NA', 'US', 'UK', 'CAN', 'GE','FRA'), rotation=0)
bar_ax.set_xlabel('')
bar_ax.set_ylabel('# tweets')

barh_ax = ax[1,0]
platXtab.loc[sourceslst,countrylst].sum(axis=1)[::-1].plot.barh(ax=barh_ax,
color=barcolors[2],**barstyle)
barh_ax.yaxis.set_label_position("left")
barh_ax.xaxis.tick_top()
barh_ax.xaxis.set_label_position("top")
barh_ax.xaxis.set_major_formatter(formatterK1)
barh_ax.set_xlim(barh_ax.get_xlim()[::-1])
```

```
barh_ax.set_yticklabels(('Hootsuite','TweetDeck','iPad','Android','iPhone','Twit
ter Web'), rotation=0)
barh_ax.set_xlabel('# tweets')
barh_ax.set_ylabel('')
plt.show()


def detect_tweetlang(tweet):
   try:
      return detect(tweet)
   except:
      return 'unknown'

# Combine two data files and drop duplicates
lang_df=pd.concat([tweets_biden[['tweet','country']].copy(),tweets_trump[['t
weet','country']].copy()])
lang_df['country'].replace({'United States of America':'United States'},
inplace=True)
lang_df=lang_df.fillna('Geo Data NA')
lang_df=lang_df.drop_duplicates()

# Randomly sample data for langauge analysis
lang_smdf=lang_df.sample(n=4000).copy()
lang_smdf['lang'] = lang_smdf['tweet'].apply(detect_tweetlang)

# Select top five languages and five countries for heatmap
langs=pd.DataFrame(lang_smdf.groupby(['lang'])['tweet'].count().sort_values(
ascending=False)[:5])
langs=langs.reset_index()
langslst=langs.lang.to_list()

country=pd.DataFrame(lang_smdf.groupby(['country'])['tweet'].count().sort_v
alues(ascending=False)[:5])
country=country.reset_index()
countrylst=country.country.to_list()

# Create a crosstab to feed data to heatmap
langXtab=pd.crosstab(lang_smdf.lang, lang_smdf.country, normalize=True)
```

```
# Identify the common UserId's in both datasets and create tables for feed
visualisation
common_ids=np.intersect1d(tweets_biden.user_id, tweets_trump.user_id)
unique_b=tweets_biden[~tweets_biden.user_id.isin(common_ids)].copy()
common_b=tweets_biden[tweets_biden.user_id.isin(common_ids)].copy()
unique_t=tweets_trump[~tweets_trump.user_id.isin(common_ids)].copy()
common_t=tweets_trump[tweets_trump.user_id.isin(common_ids)].copy()

common_df=pd.concat([common_b,common_t])
common_df=common_df.drop_duplicates()

# Create columns for visualiation
unique_b['usertype'] = 'Biden'
unique_t['usertype'] = 'Trump'
common_df['usertype'] = 'Both'

# Narrow down data
cont_df=pd.concat([unique_b[['tweet','continent','usertype']].copy(),
          unique_t[['tweet','continent','usertype']].copy(),
          common_df[['tweet','continent','usertype']].copy()])

# Label NA Geo Data
cont_df=cont_df.fillna('Geo Data NA')

# Calculate tweet counts for each usertype and continuent
usertype=pd.DataFrame(cont_df.groupby(['usertype'])['tweet'].count().sort_va
lues(ascending=False))
usertype=usertype.reset_index()
userlst=usertype.usertype.tolist()

continent=pd.DataFrame(cont_df.groupby(['continent'])['tweet'].count().sort_
values(ascending=False)[:6])
continent=continent.reset_index()
contlst=continent.continent.to_list()

# Create crosstab to feed heatmap
contXtab=pd.crosstab(cont_df.continent, cont_df.usertype)

fig, ax=plt.subplots(2,2, figsize=(5.5,9),
```

```
                  gridspec_kw={'height_ratios':[2,5], 'width_ratios':[2,3],
'wspace':0.15, 'hspace':0.1})

hide_ax = ax[0,0]
hide_axes(hide_ax)

hm_ax = ax[1,1]
draw_heatmap3(contXtab.loc[contlst,userlst], hm_ax)
hm_ax.set_xlabel('UserId Membership')
hm_ax.set_ylabel('tweets continent of origin')

bar_ax = ax[0,1]
contXtab.loc[contlst,userlst].sum().plot.bar(ax=bar_ax,
color=barcolors[7],**barstyle)
bar_ax.set_xlabel(bar_ax.get_xlabel())
bar_ax.xaxis.tick_top()
bar_ax.xaxis.set_label_position("top")
bar_ax.yaxis.set_major_formatter(formatterK1)
bar_ax.set_ylabel('# tweets')
bar_ax.set_xlabel('')

barh_ax = ax[1,0]
contXtab.loc[contlst,userlst].sum(axis=1)[::-1].plot.barh(ax=barh_ax,
color=barcolors[4],**barstyle)
barh_ax.yaxis.set_label_position("left")
barh_ax.xaxis.tick_top()
barh_ax.xaxis.set_label_position("top")
barh_ax.xaxis.set_major_formatter(formatterK1)
barh_ax.set_xlim(barh_ax.get_xlim()[::-1])
barh_ax.set_xlabel('# tweets')
barh_ax.set_ylabel('')

plt.show()

# Identify common tweet creation dates
common_creat=np.intersect1d(tweets_biden.created_at_r,
tweets_trump.created_at_r)

# Mask out data to ensure common lenth arrays to feed visualisation
```

```
cnt_tbiden=tweets_biden[tweets_biden.created_at_r.isin(common_creat)]['cr
eated_at_r'].value_counts().sort_index()
cnt_ttrump=tweets_trump[tweets_trump.created_at_r.isin(common_creat)]['
created_at_r'].value_counts().sort_index()

plt.figure(figsize=(12,5))
p6=sns.lineplot(cnt_tbiden.index, cnt_tbiden.values, color=barcolors[3],
label='Biden Dataset')
p6.set_title('Count of tweets per hour')
p6=sns.lineplot(cnt_ttrump.index, cnt_ttrump.values, color=barcolors[6],
label='Trump Dataset')
p6.set_xticks(range(0, len(cnt_tbiden.index), 24))
p6.set_xticklabels(common_df['created_at'].dt.strftime('%m-
%d').unique().tolist())
p6.set_yscale('log')
plt.show()

# Obtain tweets only from data that has Geo Data from the US
text1=tweets_biden.loc[tweets_biden['country'] == 'United States of
America']['tweet']
text2=tweets_trump.loc[tweets_trump['country'] == 'United States of
America']['tweet']

def clean1(sent):
  filtered_sent=""
  stopwords = nltk.corpus.stopwords.words('english')
  sent = (unicodedata.normalize('NFKD', sent)
      .encode('ascii', 'ignore')
      .decode('utf-8', 'ignore')
      .lower())
  sent = re.sub(r'#.+|https.+|[^(a-zA-Z)\s]','',sent)
  words=sent.split()
  for word in words:
    if word not in stopwords:
      filtered_sent=filtered_sent+' '+word
  return filtered_sent

def clean2(text):
```

```
    wnl = nltk.stem.WordNetLemmatizer()
    stopwords = nltk.corpus.stopwords.words('english')
    text = (unicodedata.normalize('NFKD', text)
            .encode('ascii', 'ignore')
            .decode('utf-8', 'ignore')
            .lower())
    words = re.sub(r'[^\w\s]', '', text).split()
    return [wnl.lemmatize(word) for word in words if word not in stopwords]

words1 = clean2(''.join(str(text1.apply(clean1).tolist())))
words2 = clean2(''.join(str(text2.apply(clean1).tolist())))

# Obtain top 10 Bi and Tri Ngrams from cleaned data
biden_2ngrams=(pd.Series(nltk.ngrams(words1, 2)).value_counts())[:10]
trump_2ngrams=(pd.Series(nltk.ngrams(words2, 2)).value_counts())[:10]
biden_3ngrams=(pd.Series(nltk.ngrams(words1, 3)).value_counts())[:10]
trump_3ngrams=(pd.Series(nltk.ngrams(words2, 3)).value_counts())[:10]

# Input Bi and Tri Ngrams into dataframes for plotting
biden_ngrams=pd.concat([biden_2ngrams,biden_3ngrams])
trump_ngrams=pd.concat([trump_2ngrams,trump_3ngrams])
fig, ax=plt.subplots(1,2, figsize=(8,16),
                gridspec_kw={'width_ratios':[1,1], 'wspace':0.1, 'hspace':0.1})

barh_ax = ax[0]
biden_ngrams[::-1].plot.barh(ax=barh_ax, color=barcolors[3],**barstyle)
barh_ax.yaxis.set_label_position("left")
barh_ax.xaxis.tick_top()
barh_ax.xaxis.set_label_position("top")
barh_ax.xaxis.set_major_formatter(formatterK2)
barh_ax.set_xlim([0, 3500])
barh_ax.set_xlim(barh_ax.get_xlim()[::-1])
barh_ax.set_xlabel('Bi & Tri N-gram Count - Biden Dataset')
barh_ax.set_ylabel('')

barh_ax = ax[1]
trump_ngrams[::-1].plot.barh(ax=barh_ax, color=barcolors[6],**barstyle)
barh_ax.xaxis.tick_top()
```

```
barh_ax.xaxis.set_label_position("top")
barh_ax.xaxis.set_major_formatter(formatterK2)
barh_ax.set_xlim([0, 3500])
barh_ax.set_xlim(barh_ax.get_xlim())
barh_ax.yaxis.tick_right()
barh_ax.set_xlabel('Bi & Tri N-gram Count - Trump Dataset')
barh_ax.set_ylabel('')
plt.show()
print("plotting the bigram and trigrams of the dataset")

import nltk
nltk.downloader.download('vader_lexicon')

# Obtain sentiment scores for both datasets

sid = SentimentIntensityAnalyzer()
tweets_biden['VADAR']=sentiment(tweets_biden['tweet'])
tweets_trump['VADAR']=sentiment(tweets_trump['tweet'])
tweets_biden['compound']  = tweets_biden['VADAR'].apply(lambda score_dict:
score_dict['compound'])
tweets_trump['compound']  = tweets_trump['VADAR'].apply(lambda
score_dict: score_dict['compound'])
tweets_trump['sentiment']  = tweets_trump['compound'].apply(lambda x:
'pos' if x > 0.05 else ('neg' if x < -0.05 else 'neu'))
tweets_biden['sentiment']  = tweets_biden['compound'].apply(lambda x: 'pos'
if x > 0.05 else ('neg' if x < -0.05 else 'neu'))

# Create 52 state set
states=set(tweets_biden.loc[tweets_biden['country'] == 'United States of
America']['state'].dropna())
states.remove('District of Columbia')
states.remove('Northern Mariana Islands')

# Create feature to allow masking of data and then mask data for votable
states
tweets_biden['voting_rights']=tweets_biden['state'].apply(lambda x: 'Yes' if x
in states else 'No')
```

```
tweets_trump['voting_rights']=tweets_trump['state'].apply(lambda x: 'Yes' if x
in states else 'No')
sent_t=tweets_trump.loc[tweets_trump['voting_rights'] == 'Yes']
sent_b=tweets_biden.loc[tweets_biden['voting_rights'] == 'Yes']

# Further mask data for only the last 14 days
state_b=sent_b.loc[sent_b['created_at'] > max(sent_b['created_at']) -
timedelta(14)]
state_t=sent_t.loc[sent_t['created_at'] > max(sent_t['created_at']) -
timedelta(14)]
state_b_mean=state_b.groupby('state')['compound'].mean().reset_index()
state_t_mean=state_t.groupby('state')['compound'].mean().reset_index()

# Further mask data for only the last 14 days
state_bp=sent_b.loc[sent_b['created_at'] < min(sent_b['created_at']) +
timedelta(14)]
state_tp=sent_t.loc[sent_t['created_at'] < min(sent_t['created_at']) +
timedelta(14)]
state_bp_mean=state_bp.groupby('state')['compound'].mean().reset_index()
state_tp_mean=state_tp.groupby('state')['compound'].mean().reset_index()

# Create dataframe for visualisation
states_sent=pd.DataFrame({'state':state_b_mean['state'],
                    'biden1':state_b_mean['compound'],
                    'trump1':state_t_mean['compound'],
                    'biden2':state_bp_mean['compound'],
                    'trump2':state_tp_mean['compound'],})

fig, ax=plt.subplots(2,1, figsize=(12,10), gridspec_kw={'hspace':0.05})
lineax=ax[0]
sns.lineplot(x='state', y='trump1', color=barcolors[6], data=states_sent,
ax=lineax, label='Trump Dataset (L14D)')
sns.scatterplot(x='state', y='trump1', color=barcolors[6], data=states_sent,
ax=lineax)
sns.lineplot(x='state', y='trump2', color='lightgrey', data=states_sent,
ax=lineax, label='Trump Dataset (F14D)')
sns.scatterplot(x='state', y='trump2', color='lightgrey', data=states_sent,
ax=lineax)
```

```
lineax.set_ylim([-0.2, 0.2])
lineax.set_ylabel('mean sentiment score (Last 14D Data)')
lineax.set_xlabel('')
plt.xticks(rotation=90)
lineax.axhline(y=0, color='k', linestyle='-')
lineax.axhline(y=0.05, color='lightgrey', linestyle='-')
lineax.axhline(y=-0.05, color='lightgrey', linestyle='-')
lineax.axes.get_xaxis().set_ticks([])
lineax.spines['right'].set_visible(False)
lineax.spines['top'].set_visible(False)
lineax.spines['bottom'].set_visible(False)

lineax=ax[1]
sns.lineplot(x='state', y='biden1', color=barcolors[3], data=states_sent,
ax=lineax, label='Biden Dataset (L14D)')
sns.scatterplot(x='state', y='biden1', color=barcolors[3], data=states_sent,
ax=lineax)
sns.lineplot(x='state', y='biden2', color='lightgrey', data=states_sent, ax=lineax,
label='Biden Dataset (F14D)')
sns.scatterplot(x='state', y='biden2', color='lightgrey', data=states_sent,
ax=lineax)
lineax.set_ylim([-0.2, 0.2])
lineax.set_ylabel('mean sentiment score')
lineax.set_xlabel('')
plt.xticks(rotation=90)
lineax.axhline(y=0, color='k', linestyle='-')
lineax.axhline(y=0.05, color='lightgrey', linestyle='-')
lineax.axhline(y=-0.05, color='lightgrey', linestyle='-')
lineax.spines['right'].set_visible(False)
lineax.spines['top'].set_visible(False)
plt.show()

# Calculate counts of sentiments
stack_t=sent_t.groupby(['created_at_r','sentiment'])['tweet'].count().reset_ind
ex()
stack_b=sent_b.groupby(['created_at_r','sentiment'])['tweet'].count().reset_in
dex()
```

```
# Setup np.arrays to allow quick calculations of the proportions of tweet
sentiments
a1=np.array(stack_b.loc[stack_b.sentiment == 'pos']['tweet'].tolist())
b1=np.array(stack_b.loc[stack_b.sentiment == 'neu']['tweet'].tolist())
c1=np.array(stack_b.loc[stack_b.sentiment == 'neg']['tweet'].tolist())
d1=np.array(stack_b.groupby('created_at_r')['tweet'].sum().tolist())

a2=np.array(stack_t.loc[stack_t.sentiment == 'pos']['tweet'].tolist())
b2=np.array(stack_t.loc[stack_t.sentiment == 'neu']['tweet'].tolist())
c2=np.array(stack_t.loc[stack_t.sentiment == 'neg']['tweet'].tolist())
d2=np.array(stack_t.groupby('created_at_r')['tweet'].sum().tolist())

# Calculate sentiment proportions and feed into dataframes for visualisation
SentiDat_b=pd.DataFrame({'date':pd.to_datetime(stack_b.created_at_r.uniqu
e()),
                'datenum':dates.datestr2num(stack_b.created_at_r.unique()),
                'pos':a1/d1,'neu':b1/d1,'neg':c1/d1})

SentiDat_t=pd.DataFrame({'date':pd.to_datetime(stack_t.created_at_r.unique
()),
                'datenum':dates.datestr2num(stack_t.created_at_r.unique()),
                'pos':a2/d2,'neu':b2/d2,'neg':c2/d2})

@plt.FuncFormatter
def fake_dates(x, pos):
   """ Custom formater to turn floats into e.g., 05-08"""
   return dates.num2date(x).strftime('%m-%d')

fig, ax=plt.subplots(3,1, figsize=(12,12), gridspec_kw={'hspace':0.05})

# Plot
lineax=ax[0]
lineax.set_title('Sentiment Analysis per Hour')
sns.regplot(x='datenum',y='pos', data=SentiDat_b, ax=lineax,
color=barcolors[3], scatter_kws={'s':5}, logistic=True, ci=95)
sns.lineplot(x='datenum',y='pos', data=SentiDat_b, ax=lineax,
color=barcolors[3], alpha=0.5, label='Biden Dataset')
sns.regplot(x='datenum',y='pos', data=SentiDat_t, ax=lineax,
color=barcolors[6], scatter_kws={'s':5}, logistic=True, ci=95)
```

```
sns.lineplot(x='datenum',y='pos', data=SentiDat_t, ax=lineax,
color=barcolors[6], alpha=0.5, label='Trump Dataset')
lineax.xaxis.set_major_formatter(fake_dates)
lineax.set_ylim([0, 0.7])
lineax.set_xlabel('')
lineax.set_ylabel('positive (proportion)')
lineax.axes.get_xaxis().set_ticks([])
lineax.spines['right'].set_visible(False)
lineax.spines['top'].set_visible(False)
lineax.spines['bottom'].set_visible(False)

lineax1=ax[1]
sns.regplot(x='datenum',y='neu', data=SentiDat_b, ax=lineax1,
color=barcolors[3], scatter_kws={'s':5}, logistic=True, ci=95)
sns.lineplot(x='datenum',y='neu', data=SentiDat_b, ax=lineax1,
color=barcolors[3], alpha=0.5)
sns.regplot(x='datenum',y='neu', data=SentiDat_t, ax=lineax1,
color=barcolors[6], scatter_kws={'s':5}, logistic=True, ci=95)
sns.lineplot(x='datenum',y='neu', data=SentiDat_t, ax=lineax1,
color=barcolors[6], alpha=0.5)
lineax1.xaxis.set_major_formatter(fake_dates)
lineax1.set_ylim([0, 0.7])
lineax1.set_xlabel('')
lineax1.set_ylabel('neutral (proportion)')
lineax1.axes.get_xaxis().set_ticks([])
lineax1.spines['right'].set_visible(False)
lineax1.spines['top'].set_visible(False)
lineax1.spines['bottom'].set_visible(False)

lineax2=ax[2]
sns.regplot(x='datenum',y='neg', data=SentiDat_b, ax=lineax2,
color=barcolors[3], scatter_kws={'s':5}, logistic=True, ci=95)
sns.lineplot(x='datenum',y='neg', data=SentiDat_b, ax=lineax2,
color=barcolors[3], alpha=0.5)
sns.regplot(x='datenum',y='neg', data=SentiDat_t, ax=lineax2,
color=barcolors[6], scatter_kws={'s':5}, logistic=True, ci=95)
sns.lineplot(x='datenum',y='neg', data=SentiDat_t, ax=lineax2,
color=barcolors[6], alpha=0.5)
```

```
lineax2.xaxis.set_major_formatter(fake_dates)
lineax2.set_ylim([0, 0.7])
lineax2.set_ylabel('negative (proportion)')
lineax2.set_xlabel('date')
lineax2.spines['right'].set_visible(False)
lineax2.spines['top'].set_visible(False)

plt.show()
```