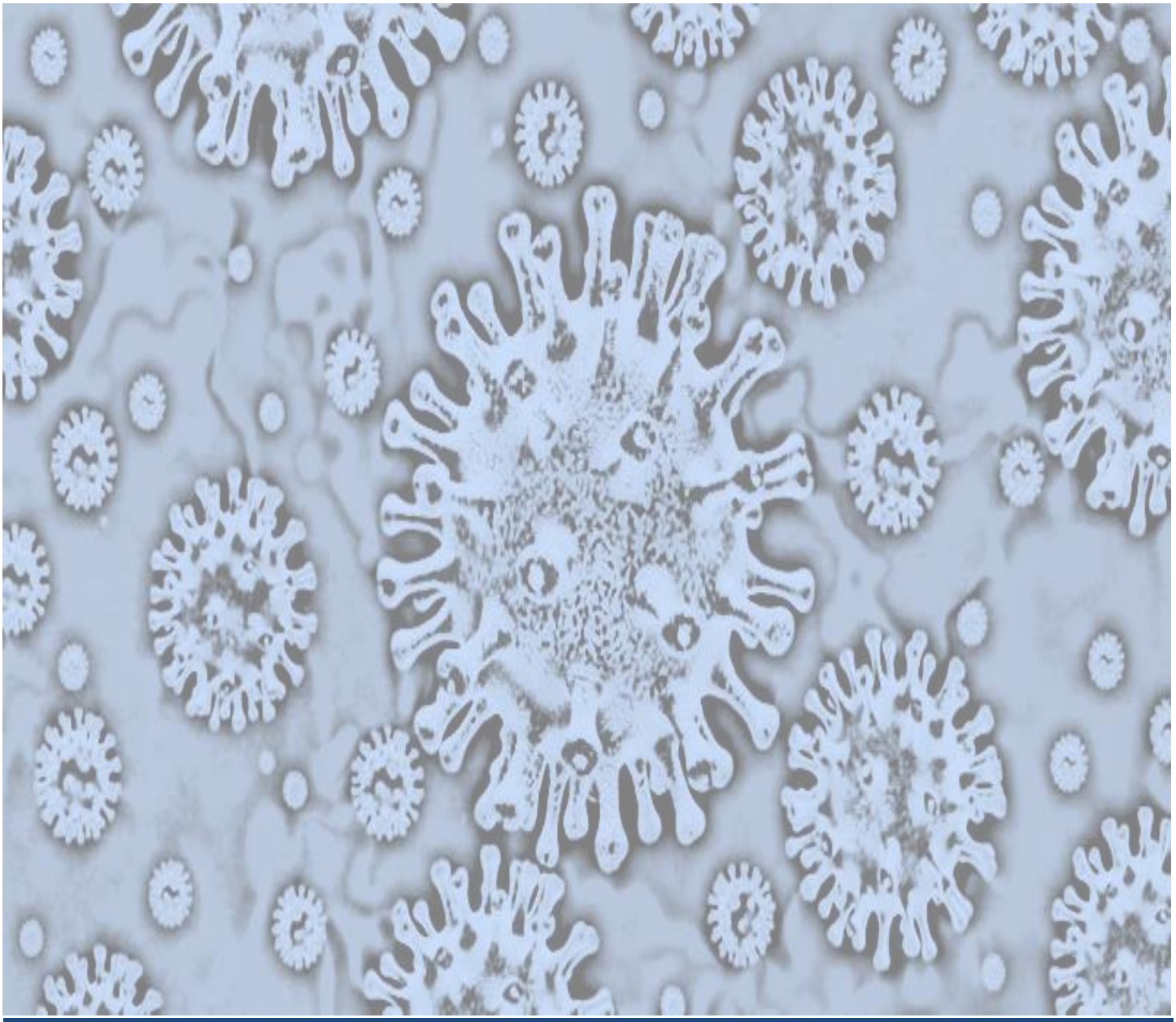


TEXT ANALYTICS PROJECT
ISM6930
COVID-19TWITTER VS JOHN HOPKINS TREND

Pankaj Patel
Indra Reddy
Srikrishna KrishnaRao
Pragati Shukla
Avinash Narra

COVID-19

TWITTER VS JOHN HOPKINS TREND

Contents

Executive Summary:

How sad is that the people who remember the last major pandemic influenza in 1968 are the primary victims of todays. How sad that despite many medical advances that have been made since then, the treatments offered to many patients in areas where Covid-19 has exploded are the same ones they might have received in that era. Perhaps the lessons they remember, those of quarantine, isolation, and social distancing, are the ones that will save us again.

The earliest cases were identified through pneumonia surveillance mechanism. The infection is thought to have originated in a food market in the Chinese city of Wuhan. It is transmitted to people by direct contact or respiratory droplets from infected people. The experiments suggest that viruses in the lineage are ready to jump to humans directly from bats. Clinical studies of hospitalized patients frequently show symptoms associated with viral pneumonia, most commonly fever, cough, sore throat, myalgia, and fatigue. Currently there is no medicine or vaccine available to cure or prevent Covid-19. Face masks block the spread of coronaviruses in respiratory droplets, suggesting that masks could prevent the transmission of coronaviruses. Infection containment and personal protection are the most important measures to prevent infections and contain viral spread. In the United States, disease surveillance is supported by the CDC Division of Health Informatics and Surveillance and is carried out through a variety of networks that involve the collaboration of thousands of agencies at the federal, state, territorial levels across various health departments. Importantly, cases reported from CDC are of high quality, such reporting is not timely due to internal protocol of these offices to collect and verify data prior to formal publication. So, to collect information with minimal delay, we have opted John Hopkin's University Website.

There are a lot of speculations on the credibility of John Hopkins data. Among the social media platforms, twitter has the most reliable information. So, we have adopted the twitter as our basic source of information for the coronavirus tweets. After a lot of preprocessing and analyzing the tweets, we have found the correlation between John Hopkins data and the twitter data in some of the states, but the others seem to have a different trend.

Problem Definition and Significance:

A lot of media reports have been published about the credibility of John Hopkins real time data. So, our analysis tries to verify the credibility of John Hopkins data on the COVID-19. In our study, we demonstrate the correlation between the number of tweets and actual cases i.e. both the affected and deceased. We have also assessed whether twitter data is used to predict positive cases.

Corona cases reported from official sources like John Hopkins Coronavirus Resource Center are of high quality and such reporting is not timely due to internal protocol of verifying the data prior to publication. In addition, the cases reported by any single source don't always showcase

the real numbers. As of on April 21st, there are approximately 806,100 confirmed cases and 45,100 deaths in the U.S as per John Hopkins data center.

Prior Literature:

The escalating outbreak has prompted a flurry of research activity on the coronavirus, which emerged in December and is new to Science. Several dozens of papers have been published on how rapidly the virus spreads, or the length of the incubation period. The incubation period for corona virus is estimated to be 14 days. So, any model developed for the prediction of coronavirus should take the incubation period into account.

The first paper was on improving the surveillance of Zika virus in the countries with poor infrastructure and well-established systems often do not exist. For their experiments, they have chosen the immediately available time and geo-tagged data from twitter. They have extracted the tweets using keywords like zika and mosquito. They have compared the CDC real time figures with the ZIKV twitter data. Time series Analysis was conducted for weekly ZIKV cases and zika tweets to illustrate their patterns. The cumulative prevalence was also examined and correlated with cumulative zika tweets spatially all over the U.S. They have examined the model using 0-6 weeks of lag which considers 1-2-week incubation period of ZIKV as well as the potential 2-4-week delay between ZIKV laboratory testing and reporting. They have significantly predicted ZIKV cases using auto-regression model 1 week in advance in almost all the states with reasonable accuracy.

The second paper was on the Fake news that was circulating during Covid-19. One of the scientists at Italy's Bruno Kessler Foundation's center for Information and Communication Technology has traced these online rumors and bots that spread false news. Their experimental results show that how misinformation about coronavirus is going viral at a disturbing rate. With the flood of social media messages about the coronavirus pandemic, experts found that the mood of whole world is grim. He along with his colleagues recently analyzed more than 121,407,000 tweets, almost 50% of which were tagged to a location. The team also worked on more than 22 million Web Pages to find out the messages people and robots have been sending during the outbreak. Some of the major viral claims are that Chinese scientists created the coronavirus in a laboratory, drinking bleach or eating garlic cures the infection, Pope Francis caught Covid-19 and 5G technology caused the sickness. Of all the countries suffering worldwide, the data suggests that Iranian twitter users were most exposed to false information from social media. In February, 73 Iranians died, and hundreds were hospitalized after drinking toxic alcohol following rumors which claimed that drinking alcohol would prevent from being affected by coronavirus. South Korea was the least affected.

Data Source/Preparation:

Our source of data is twitter from which we have extracted the tweets using Tweepy library based on keywords like Wuhan virus, covid, corona-virus, covid20, coronavirus, #coronavirus, #coronaupdate, #covid19, #covid-19, covid-19, #sarscov2, #hcov19, #quidprocovid19, @CDC, hydroxychloroquine, #stayhome, #StayHome, azithromycin, #staysafe, #breakthechain, #istayhomefor, #21daysLockdown, #QuarantineLife, #StaySafe, #FlattenTheCurve, #Lockdown, test positive for our analysis. We have collected the real time figures of the affected from John Hopkins University website and compared these actual numbers with our predicted numbers which we got from our analysis.

The tweets collected were then cleaned and preprocessed for analysis. Text was normalized by performing by tokenizing the words, removing the special characters (except hashtags# and handles@ which hold useful information), removing the stop words, removing http URLs, lemmatizing and filtering the words based on the parts of speech tag. Additional to punctuations and special characters, many tweets contained emojis. Emoji's Unicode's were leveraged to handle them during the text-cleaning. After multiple rounds of text cleaning following is an example of tweet before and after cleaning

```
'The people screaming about "UKRAINE!" throughout Jan. are telling us they were way ahead of the curve on #WuhanVirus 🙄\r\n\r\nLeave it to WaPo to fall back
ws playbook -- unnamed hacks in the intel community making baseless claims https://t.co/ooj0wq4gXb'
```

After cleaning:

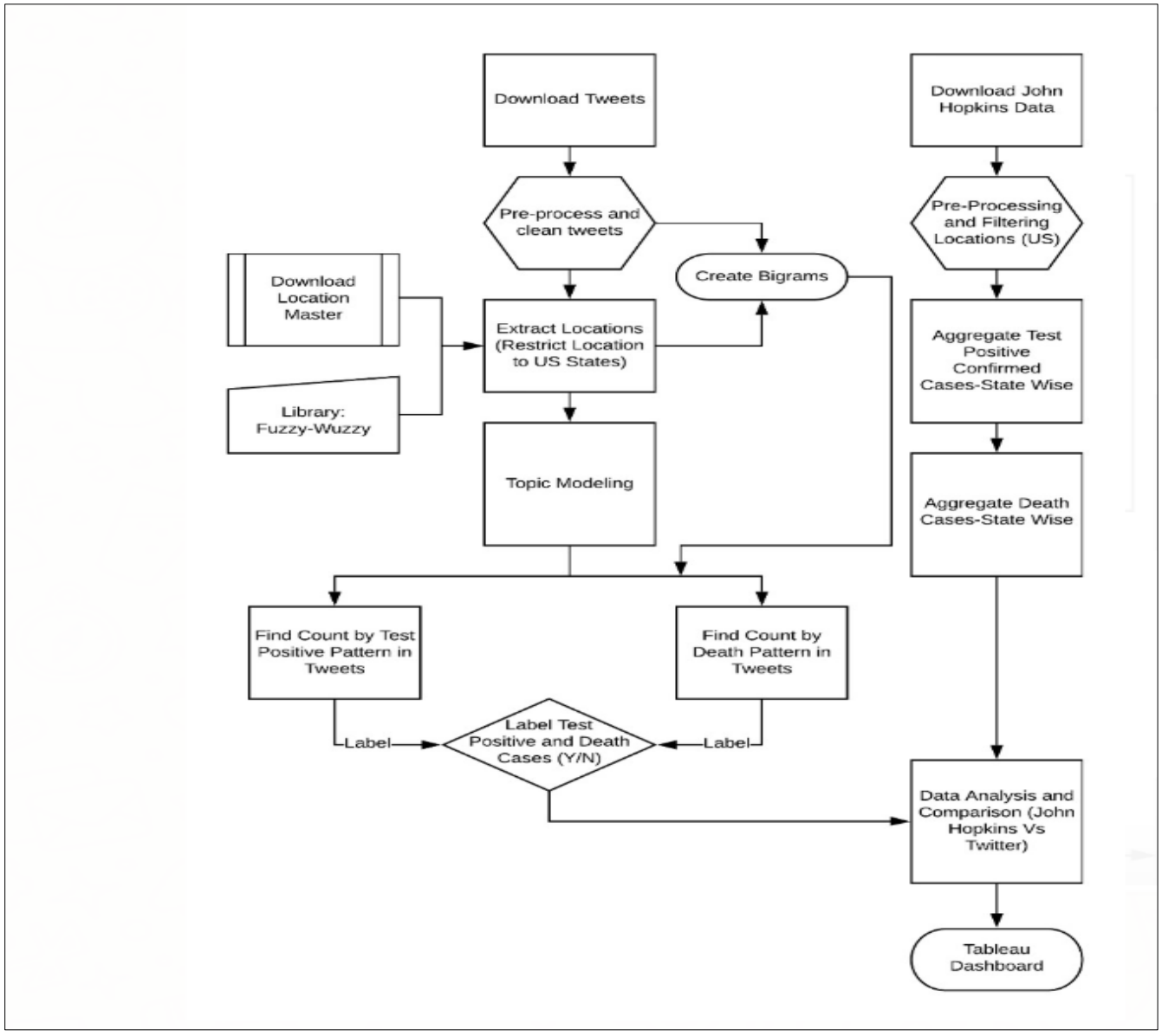
```
'people ukraine jan way #wuhanvirus fall fake news playbook unnamed hack intel community baseless claim'
```

Text Analytics Workflow:

Our analysis started with downloading tweets using our formulated keywords from twitter daily. We have collected and aggregated the daily twitter data for one month; with the restriction on language and differential usage of twitter across the regions limited us to analyze the data only in the United States. Feature selection is done based on our analysis and a lot of preprocessing and standardization is done on those selected columns. We have selected features like date, country, state, tweet, etc. Preprocessing like stop word removal, stemming, lemmatizing, removing hyperlinks and special characters, date formatting is done. With the tokenized tweets, we have created bi grams for tweets and locations. Some of the top bigrams we have created are test-posit, stay-home, hydroxychloroquine, etc. Location is one of the important features of our analysis. It's not straightforward as we think we got from the twitter. For extracting the locations from the tweets, we have used location master data file as a reference or dictionary. Fuzzy-wuzzy library is used to standardize the locations. Filtration is done to get the tweets only from U.S. The bi grams are then used to identify whether the tweet has the tendency to show corona positivity. After finding the tweet context, we have labelled them as positive or negative. Finally,

we get the number of total positive cases based on our analysis. The next part of our analysis is to compare these numbers with the real time figures of John Hopkins Data. For that purpose, we have also cleaned and aggregated John Hopkins Data and put them state-wise. We then compared the trends between John Hopkins and the twitter data. In our analysis, some of the states like New Jersey, Illinois and Texas showed similarity with John Hopkin's real time data.

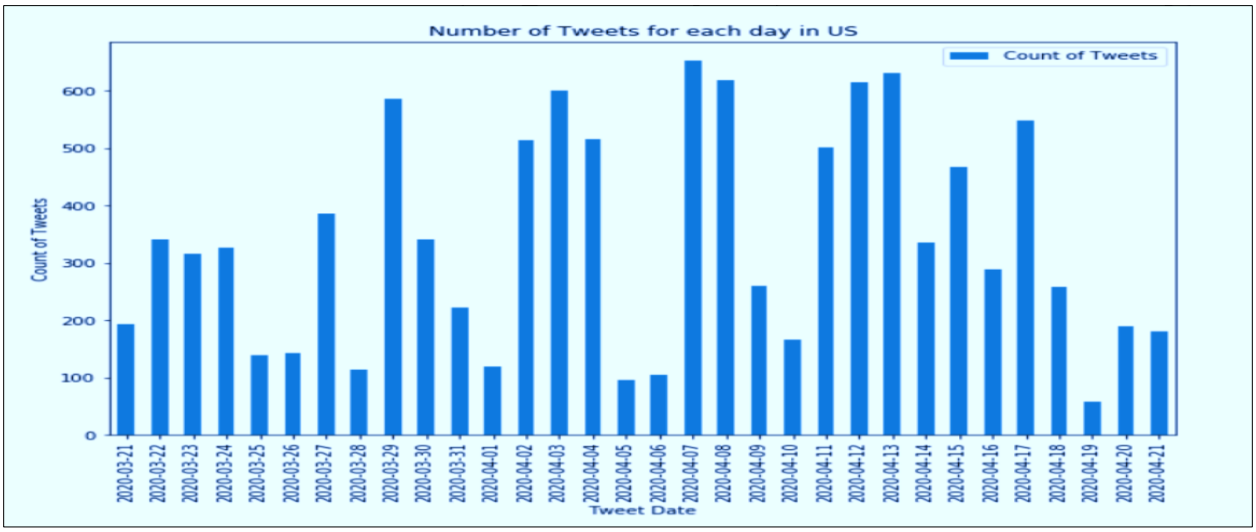
FLOWCHART



Exploratory Data Analysis:

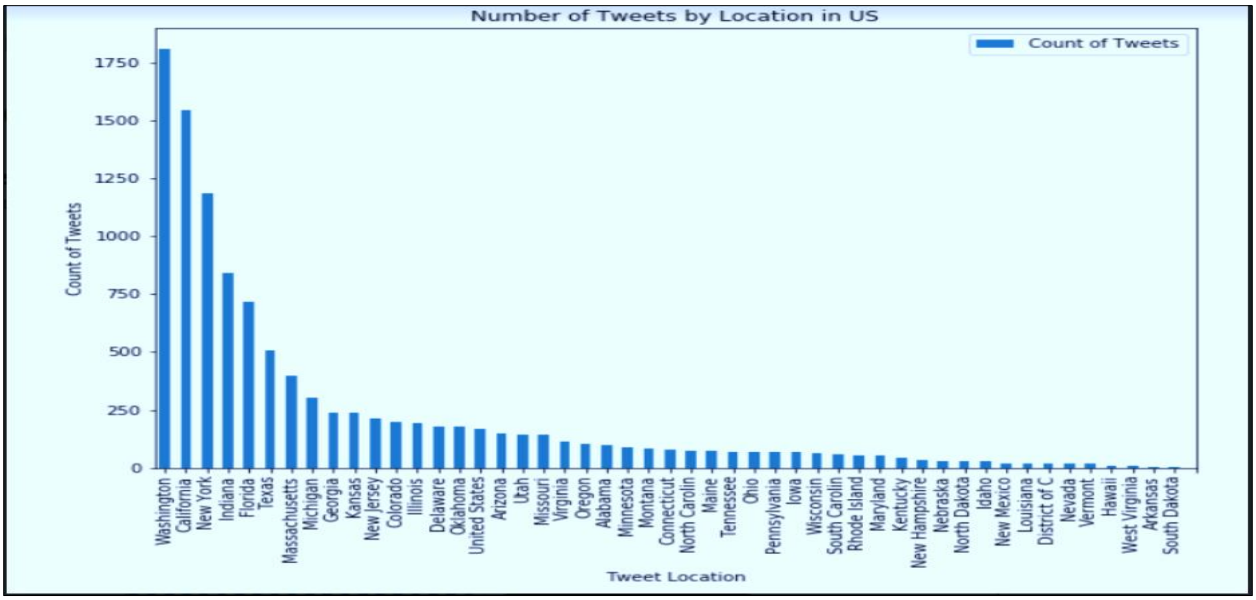
We have done a lot of exploratory data analysis. We have almost 11k tweets from United States after we have filtered by country in the time period. First, we have explored the tweet count by date and observed the trend how the people in the U.S are reacting on the coronavirus pandemic. We observed a lot of fluctuations on the reactions of people using twitter. This happens generally in twitter based on the retweeting a viral tweet or it depends on the number of people active on twitter on that day. Next we have done the analysis on the number of tweets by state in the United States. The top tweeting states are Washington, California, New York, Indiana, Florida, etc. This kind of analysis gives us better results because the number of tweets per 100,000 people on an average will be high for some states when compared to other states. So, the analysis is done based on the tweet volume. Our next part of analysis deals with finding the bi grams which are mostly used by the people in their tweets to indicate any sort of coronavirus infection or confirmation. The top tweeted bi-grams which we thought they could help in finding the positive cases are test-posit, stay-home, hydroxychloroquine-azithromycin, stayhome-staysafe, social-distance, etc.

1.Count of tweets for each day throughout USA from 21st March to 21st April.



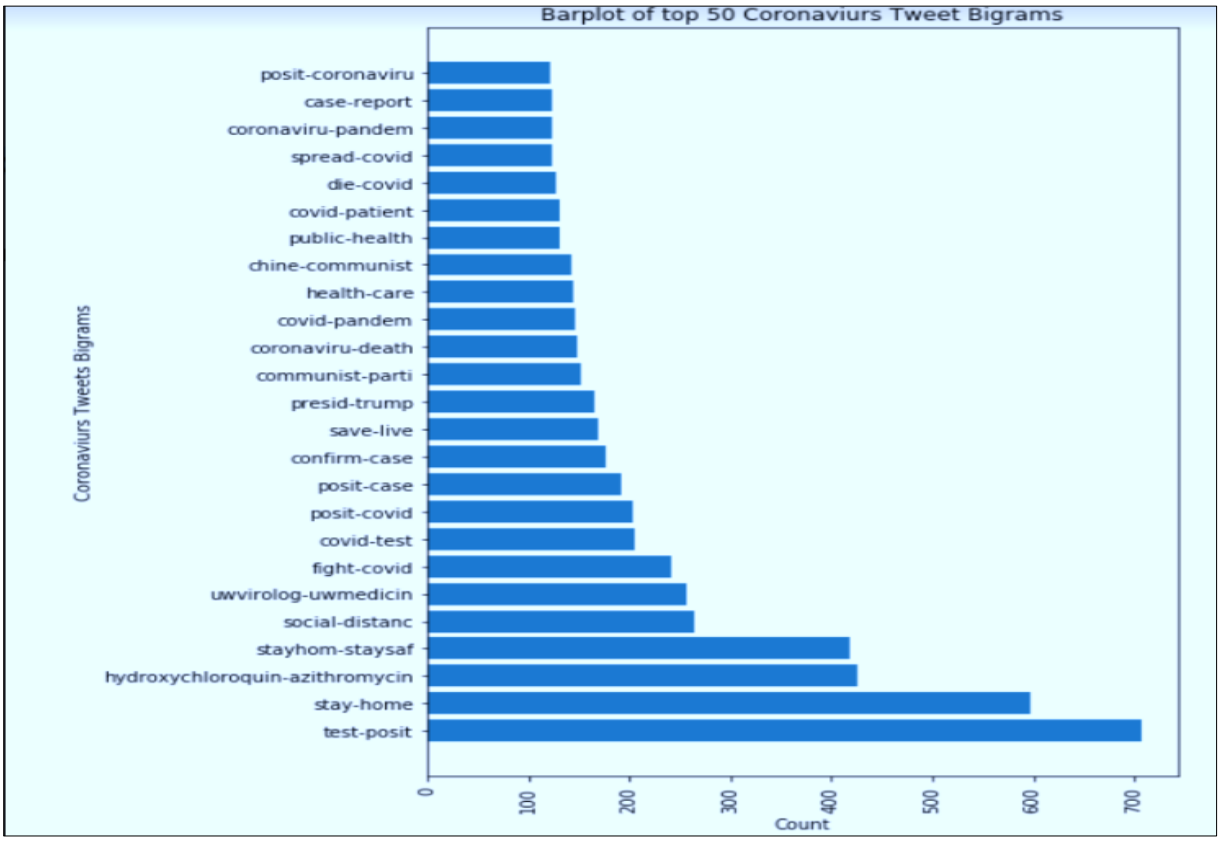
To get an idea about the overall tweet counts in US. Higher tweet counts can mean higher infected cases. Within these dates, total number of tweets were 16,328 worldwide. USA alone had 10,820 tweets. This shows that people in USA are more likely to talk about the issue on social media.

2.Tweets count plot by US states.



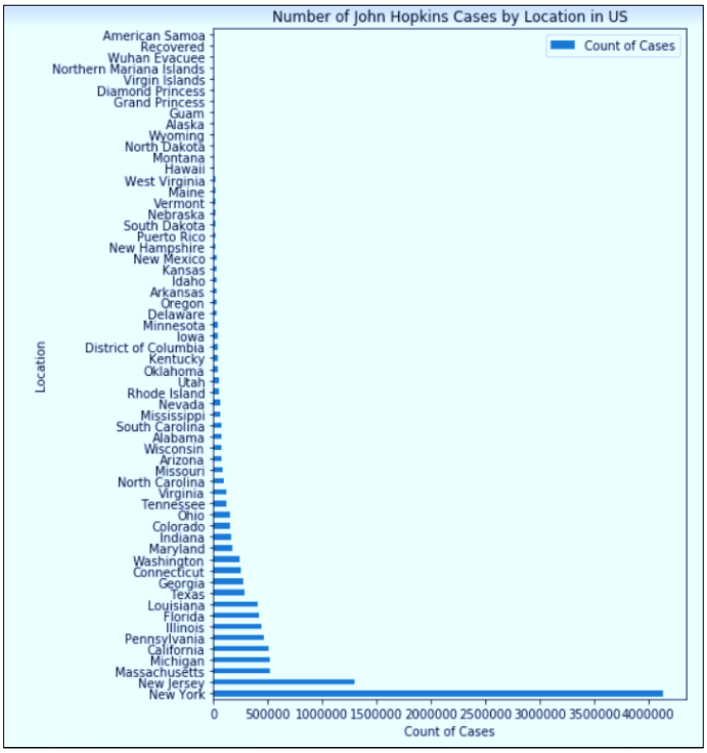
It is interesting to note that the highest tweeting states have the highest number of COVID-19 cases as per John-Hopkins.

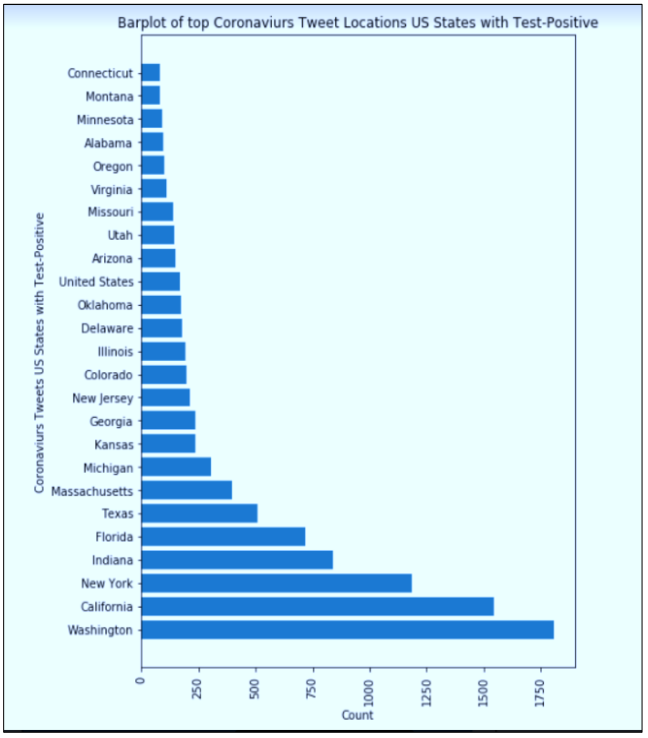
3.Barplot of top 50 Coronavirus tweet bigrams.



The bi-grams like posit-coronavirus, die-covid, test-posit helped in identifying keywords which were used to label data.

4.Barplots of top COVID-19 confirmed cases -TWITTER vs John-Hopkin





As per John-Hopkins data, states like New York, New Jersey and Massachusetts have highest COVID cases.

As per tweets, Washington, California and New York are states with highest COVID cases.

The twitter positive cases count keeps going up and down. But the positive cases as per john-Hopkins show an increasing trend.

Topic Modeling:

As an initial step topic modeling is performed to find main topics from overall tweets. LDA modeling technique is used to explore the topics. Following are the topics by using CountVectorizer:

```
In [314]: lda_DTM.print_topics()
Out[314]: [(0,
  '0.018*"people" + 0.011*"death" + 0.009*"#staysafe" + 0.009*"&amp;" + 0.009*"virus" + 0.008*"number" + 0.008*"#wuhanvirus" + 0.007*"country" + 0.006*"thing" + 0.006*"healt
h"'),
 (1,
  '0.019*"case" + 0.014*"#stayhome" + 0.012*"death" + 0.011*"virus" + 0.011*"new" + 0.009*"trump" + 0.008*"today" + 0.006*"state" + 0.006*"corona" + 0.006*"medical"'),
 (2,
  '0.015*"test" + 0.013*"positive" + 0.012*"time" + 0.011*"patient" + 0.010*"world" + 0.010*"#wuhanvirus" + 0.009*"home" + 0.008*"china" + 0.007*"chinese" + 0.007*"healt
h"'),
 (3,
  '0.011*"azithromycin" + 0.011*"hospital" + 0.010*"virus" + 0.008*"state" + 0.007*"doctor" + 0.007*"people" + 0.007*"work" + 0.006*"patient" + 0.006*"health" + 0.006*"hom
e"')]

Topic0: Caution
Topic1: Lock Down- StayHome
Topic2: Cases Tested Positive
Topic3: Medicines
```

Following are the topics using TFIDF by tuning hyperparameters:

```
In [318]: lda_tfidf.print_topics()
Out[318]: [(0,
  '0.006*"death" + 0.005*"case" + 0.004*"number" + 0.003*"new" + 0.003*"china" + 0.003*"patient" + 0.003*"#sarscov2" + 0.003*"total" + 0.002*"people" + 0.002*"day"'),
 (1,
  '0.004*"positive" + 0.004*"home" + 0.004*"test" + 0.004*"#stayhome" + 0.004*"stay" + 0.003*"time" + 0.003*"people" + 0.003*"#wuhanvirus" + 0.003*"#stayhealthy" + 0.003*"vi
rus"'),
 (2,
  '0.003*"#stayhome" + 0.003*"mask" + 0.003*"spread" + 0.003*"new" + 0.002*"people" + 0.002*"health" + 0.002*"fight" + 0.002*"today" + 0.002*"time" + 0.002*"india"'),
 (3,
  '0.003*"virus" + 0.002*"time" + 0.002*"patient" + 0.002*"help" + 0.002*"people" + 0.002*"health" + 0.002*"country" + 0.002*"everyone" + 0.002*"ventilator" + 0.002*"huma
n"')]

In [ ]: Topic0: Death Metrics
Topic1: Cases Tested Positive
Topic2: Lock Down- StayHome
Topic3: Caution
```

From the above results, it can be viewed that Cases tested positives as a distinct topic in the tweets and also Death metrics. Hence twitter data is assumed to have correlation with actual confirmed cases and deaths which is analyzed and discussed in the further steps.

How Fuzzy-Wuzzy works (it works like Similarity Matrix):

All the major cities and counties in US are downloaded as Location Master. To this location master, all the major populated cities in the world are appended. Total of 548 individual location lines are added.

Following function is called in a loop over all the 16000 tweet locations and State location master records for ‘State’ and 548 location master records for ‘City/County’

Fuzz.ratio_State = Fuzz.ratio(‘Location Master State’, ‘Tweet Location’)

Fuzz.ratio_City/County=Fuzz.ratio(‘Location Master City/County’, ‘Tweet Location’)

The greater of the two ratio is the closer match. It is stored as the matching State and Country. This way, the locations were standardized for 16000 tweets without manual intervention.

Fuzz.score() is nothing but a Similarity Score based system. It assigns a score based on how similar the two words are.

How Patten Match works:

A Pattern string is created for Positive Cases comprising of the following:

Test Posit | Posit Covid | Covid Posit | Corona Posit | Posit Corona | Case Posit | Posit Case

These intuitions were received based on Bi-gram analysis of the Cleaned Tweet Text and by plotting the Top 25 Bigrams.

Each of the 16000 Tweet Clean-text is matched against this combined pattern using Pandas Dataframe library. If a tweet clean-text matches any of the pattern, it is classified as Test-Positive-Y/N = True, else it is classified as Test-Positive-Y/N = False

Similarly, Covid Death | Death is used as the pattern match to classify each tweet.

Upon receiving the True/False for each tweet, now it is possible to group by this column by State and Date to get the daily total positive cases.

Location Classification Precision, Recall, F1 Score:

A random sample of 50 locations is selected and manually labeled. Then using the Predicted location by the above logic, Confusion Matrix, Precision, Recall and F1 Score were calculated. Following are the results:

Confusion Matrix:

	Predicted	
Actual	TP – 27	FN – 23
	FP – 0	TN-0

Precision: 0.54
Recall: 0.46
F1 Score: 0.51

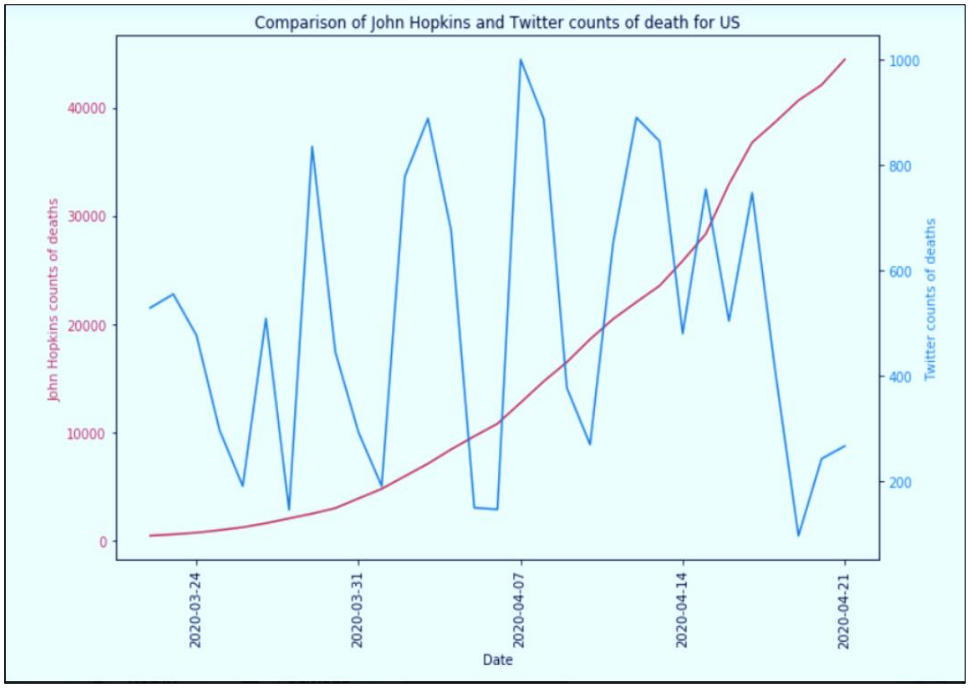
Choice and Rationale for Text Analytic Methods and Results:

One of the main challenges of this analysis was faced during preprocessing of incoming twitter live data. The location columns obtained from geo-tagged tweets was messy. To get the exact location of the tweet, we have used location master and fuzzy-wuzzy library. Location master acts a reference as it contains all the cities and states by country across the world. The fuzzy-wuzzy package helps in standardizing the location for all the tweets automatically. After we got the cleaned locations, we have filtered the tweets by country and limited our analysis to U.S.

Our goal was to identify positive cases using the tweets. First, we have manually sampled the tweets using our formulated keywords which we thought of helping us in finding the positivity of coronavirus. After doing a lot of research we have manually created the keyword list. We have created a pattern string with the keywords. After that for the classification of tweets, we have adopted bi-gram technique rather than a single word analysis because the context can then be extracted more clearly from the tweets. Each tweet is searched for the pattern string and labelled as corona positive or not. We have adopted the same technique for finding the number of deaths with different set of keywords. We have also done sentiment analysis all over U.S to verify whether the classified tweets are leading us to the correct direction or not.

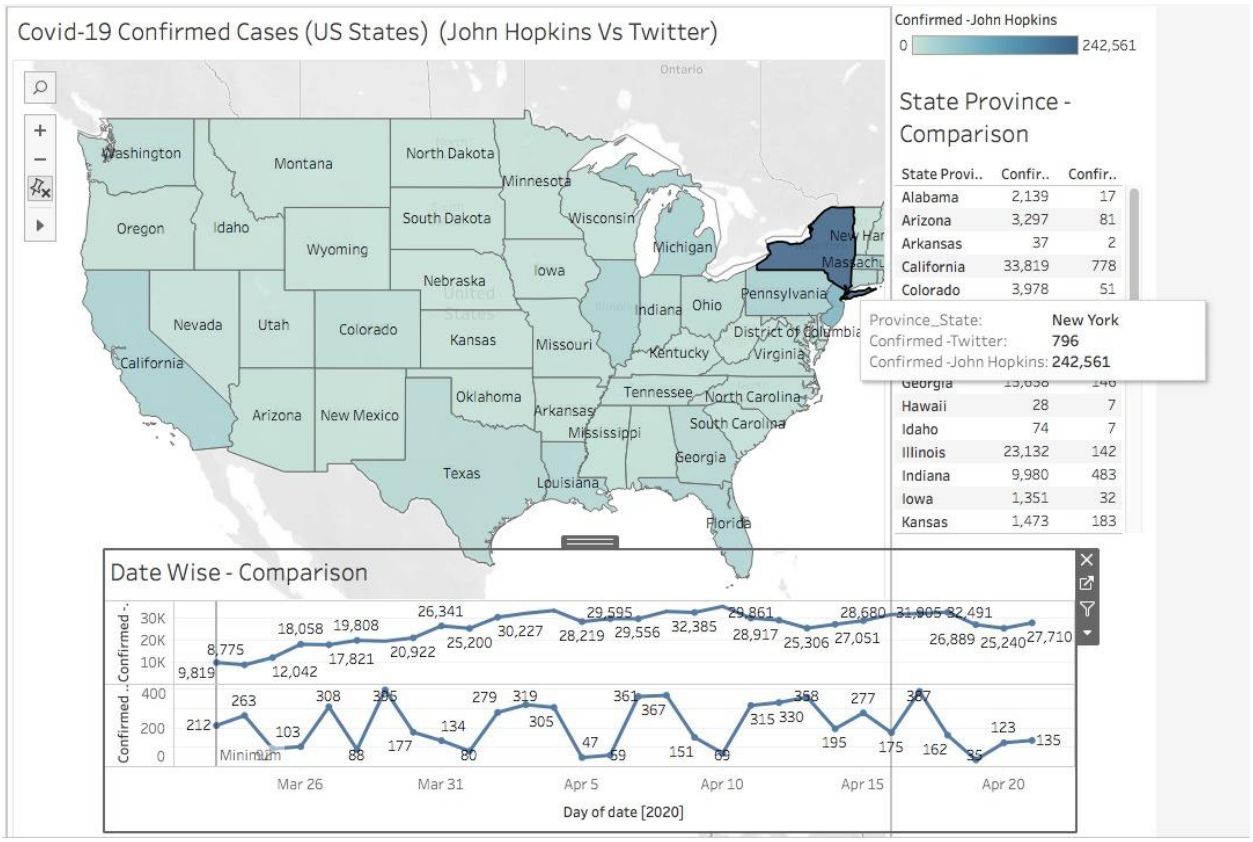
Comparing the actual cases with twitter count overall seems to mislead if we generalize it to the states as well. As the twitter usage and the number of tweets related to coronavirus are of different volume in different states, we have adopted state-wise analysis and found the patterns were different for each state. Some of the states showed similar behavior with country wise pattern and some didn't. Some of the high tweeting states like California, Florida showed abrupt trend and seems not to follow or correlate with the actual figures.

Following is the comparison of John-Hopkins vs Twitter data count of death cases:



The twitter death counts keep going up and down. But the death cases as per john-Hopkins show an increasing trend

Comparisons between both the data sources are being displayed on the Tableau dashboard. State wise figures can be visualized to find the comparisons metrics for each state. Following shows the snapshot of the dashboard:



Dashboard can be accessed here:
https://public.tableau.com/profile/pankaj.patel#!/vizhome/Covid-19USState-ConfirmedCasesJohnHopkinsVsTwitter/Dashboard1?publish=yes

Conclusion:

Out of the top 6 locations that were compared, 4 of them matched with overall trend (though daily trends had up and down swings in Twitter data). These up and down swings are presumed to be attributed to the tweeting penchant of the users, sometimes they flare up and tweet more and when things are relatively normal, they tweet less. However, the overall trend matched for Washington, Florida, Indiana and New Jersey. They did not match for New York and Texas. Overall trend for US matched as well. As New York is considered the epicenter of this coronavirus pandemic and also they had media reports such as adding 3000+ cases on a single day, giving further doubts about their numbers, it is highly likely that our twitter analysis is accurate in identifying that the trend for New York given in John Hopkins data is not very reliable and require further study.

Insights and Recommendations:

The key insights of our analysis are almost all the states exhibit both temporal correlations i.e. our twitter analysis followed the trend of John Hopkins data with respect to time and spatial

correlation i.e. the twitter count with respect to particular state also followed the trend of John Hopkins data. These correlations can be observed from our dashboard developed in tableau.

One of main recommendations is to include the 0-4 weeks lag between twitter analysis and the John Hopkins data. This lag is including the incubation period of coronavirus and testing period of the infected person. As the data goes on increasing, we can observe a lot of spikes in both Twitter count and John Hopkins data. So, the inclusion of lag helps in observing better results. Another recommendation is the inclusion of the prediction model, which helps in forecasting the future cases. This can be very helpful to warn or alert the governments and take the necessary precautions.

References:

[1] https://bmcpublichealth.biomedcentral.com/articles/10.1186/s12889-019-7103-8 - Sec2

[2] https://towardsdatascience.com/fuzzy-string-matching-in-python-68f240d910fe

[3] https://www.washingtonpost.com/science/2020/03/17/analysis-millions-coronavirus-tweets-shows-whole-world-is-sad/

[4] https://members.tortoisemedia.com/2020/03/23/the-infodemic-fake-news-coronavirus/content.html?sig=kidXw_lEkCKAedQIRVIyXO1LFE3_xVcTKtUB2-bZ35A

Appendix:

Tableau Dashboard

https://public.tableau.com/profile/pankaj.patel#!/vizhome/Covid-19USState-ConfirmedCasesJohnHopkinsVsTwitter/Dashboard1?publish=yes

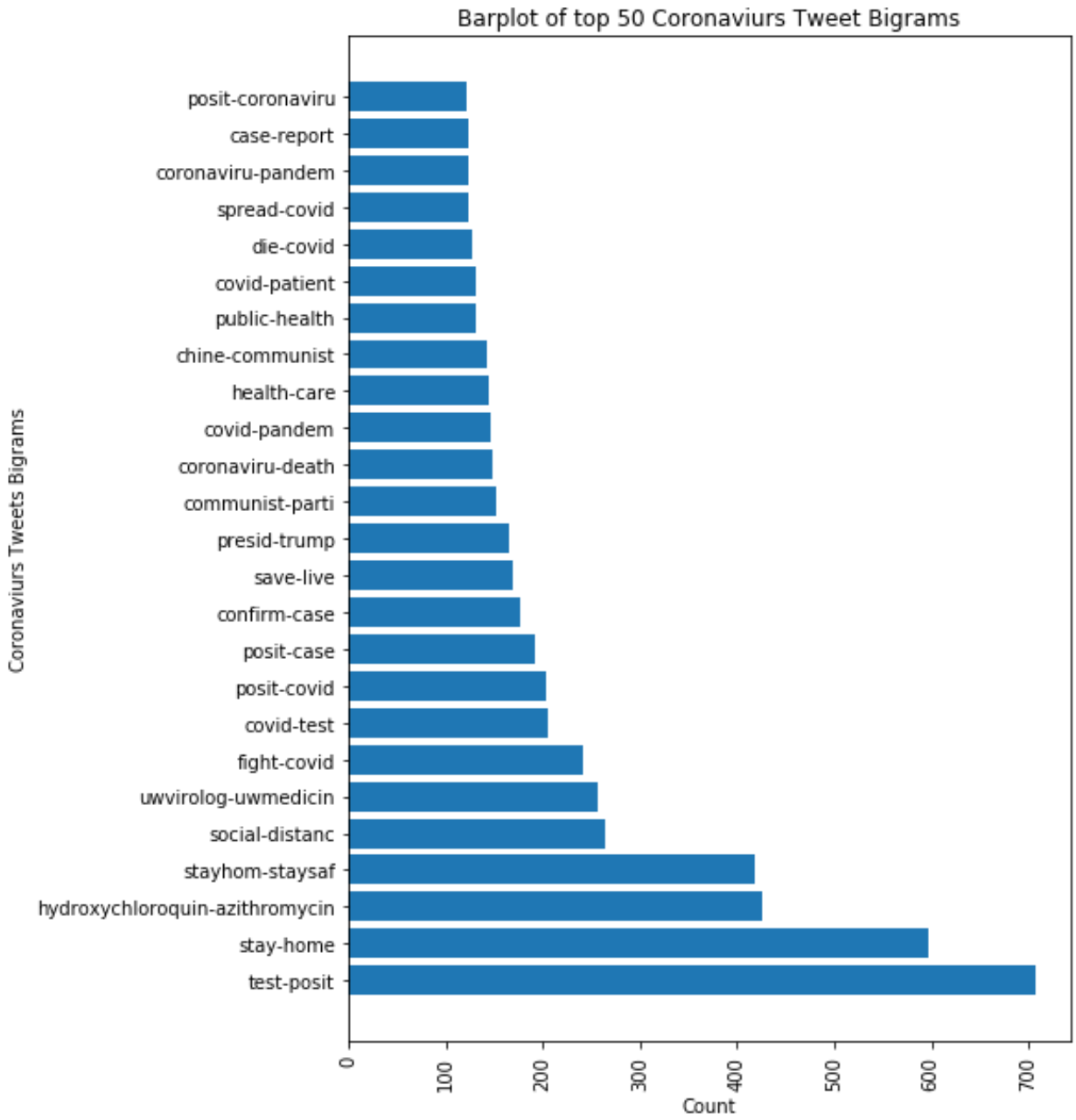
Code:

```
import pandas as pd
import os
import numpy as np
from collections import Counter
from string import printable
from string import punctuation
import unicodedata

from gensim import corpora, models, similarities                    # LDA model
from gensim.similarities import MatrixSimilarity

import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

```
from nltk import word_tokenize, pos_tag
from nltk.util import ngrams

from fuzzywuzzy import fuzz

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer       # TF-IDF
Vectorizer for LSA model
from sklearn.decomposition import TruncatedSVD                    # Singular
Value Decomposition for LSA model

import matplotlib.pyplot as plt
import seaborn as sns
C:\Users\brindhasrikrishna\Anaconda2\lib\site-packages\fuzzywuzzy\fuzz.py:11:
UserWarning: Using slow pure-python SequenceMatcher. Install python-
Levenshtein to remove this warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-
Levenshtein to remove this warning')
def data_preprocessing(df, col_name, output_col_name_1, output_col_name_2):
    st= set(printable)
    df[output_col_name_1] = df[col_name].str.lower()
    df[output_col_name_1] = df[output_col_name_1].str.strip(punctuation)
    df[output_col_name_1] = df[output_col_name_1].str.replace(r"[\"\'\,!]",
''')
    df[output_col_name_1] =
df[output_col_name_1].str.replace(r'https?://[^\s<>"]+|www\.[^\s<>"]+', "")
    df[output_col_name_1] = df[output_col_name_1].str.replace("\r\n\r\n", '')
    df[output_col_name_1] = df[output_col_name_1].str.replace("\r\n", '')
    df[output_col_name_1] = df[output_col_name_1].astype(str).fillna('No
location')
    df[output_col_name_1] = df[output_col_name_1].apply(lambda x: ''.join(["
" if i not in st else i for i in x]))
    df[output_col_name_1] = df[output_col_name_1].str.encode('ascii',
'ignore').str.decode('ascii')
    df[output_col_name_2] =
df[output_col_name_1].str.findall('\w{4,}').str.join(' ')

    return(df)
def create_tokens_stem_lem_bigrams(df, col_name, output_col_name_1,
output_col_name_2):
    porter = nltk.stem.PorterStemmer()
    lemmatizer = WordNetLemmatizer()
    stop_words = nltk.corpus.stopwords.words('english')
    stop_words.extend(['corona', 'virus', 'sarscov2', 'coronavirususa',
'hcov19', 'covid19'])
    df[output_col_name_1] = df[col_name].apply(word_tokenize)
    df[output_col_name_1] = df[output_col_name_1].apply(lambda x: [w for w in
x if w not in stop_words ])
    df[output_col_name_1] = df[output_col_name_1].apply(lambda x:
[porter.stem(w) for w in x])
    df[output_col_name_1] = df[output_col_name_1].apply(lambda x:
[lemmatizer.lemmatize(w) for w in x])
    df[output_col_name_2] = df[output_col_name_1].apply(lambda x:
list(ngrams(x,2)))

    return(df)
def create_bigram_frequency(bigrams_list):
```

```
    bigrams_dict = {}
    for bigrams in bigrams_list:
        for bigram in bigrams:
            bigrams_dict[bigram] = bigrams_dict.get(bigram, 0) + 1
    bigrams_freq = []
    for key, value in bigrams_dict.items():
        bigrams_freq.append((value, key))

    return(bigrams_freq)
def create_location_frequency(location_list):
    location_dict = {}
    for location in location_list:
        location_dict[location] = location_dict.get(location, 0) + 1
    location_freq = []
    for key, value in location_dict.items():
        location_freq.append((value, key))

    return(location_freq)
def count_occurrence_of_words(df, input_col, output_col_1, output_col_2,
pattern):
    df[output_col_1] = [','.join(map(str,l)) for l in df[input_col]]
    df[output_col_2] = df[output_col_1].str.contains(pattern)

    return(df)
def standardize_location(df, df_loc, data_location_col, loc_master_state_col,
loc_master_city_col,\
                         loc_master_country_col, output_state_col, \
                         output_country_col):
    loc_state = []
    loc_country=[]
    for i in df.index:
        max_ratio=0
        max_ratio_state=''
        max_ratio_country=''
        for j in df_loc.index:
            fr_state = fuzz.ratio(df.loc[i, data_location_col], df_loc.loc[j,
loc_master_state_col])
            if fr_state > max_ratio:
                max_ratio = fr_state
                max_ratio_state = df_loc.loc[j, loc_master_state_col]
                max_ratio_country=df_loc.loc[j, loc_master_country_col]

            fr_city = fuzz.ratio(df.loc[i, data_location_col],
df_loc.loc[j,loc_master_city_col])
            if fr_city>max_ratio:
                max_ratio = fr_city
                max_ratio_state = df_loc.loc[j, loc_master_state_col]
                max_ratio_country=df_loc.loc[j, loc_master_country_col]

        loc_state.append(max_ratio_state)
        loc_country.append(max_ratio_country)

    df[output_state_col] = loc_state
    df[output_country_col] = loc_country

    return(df)
```

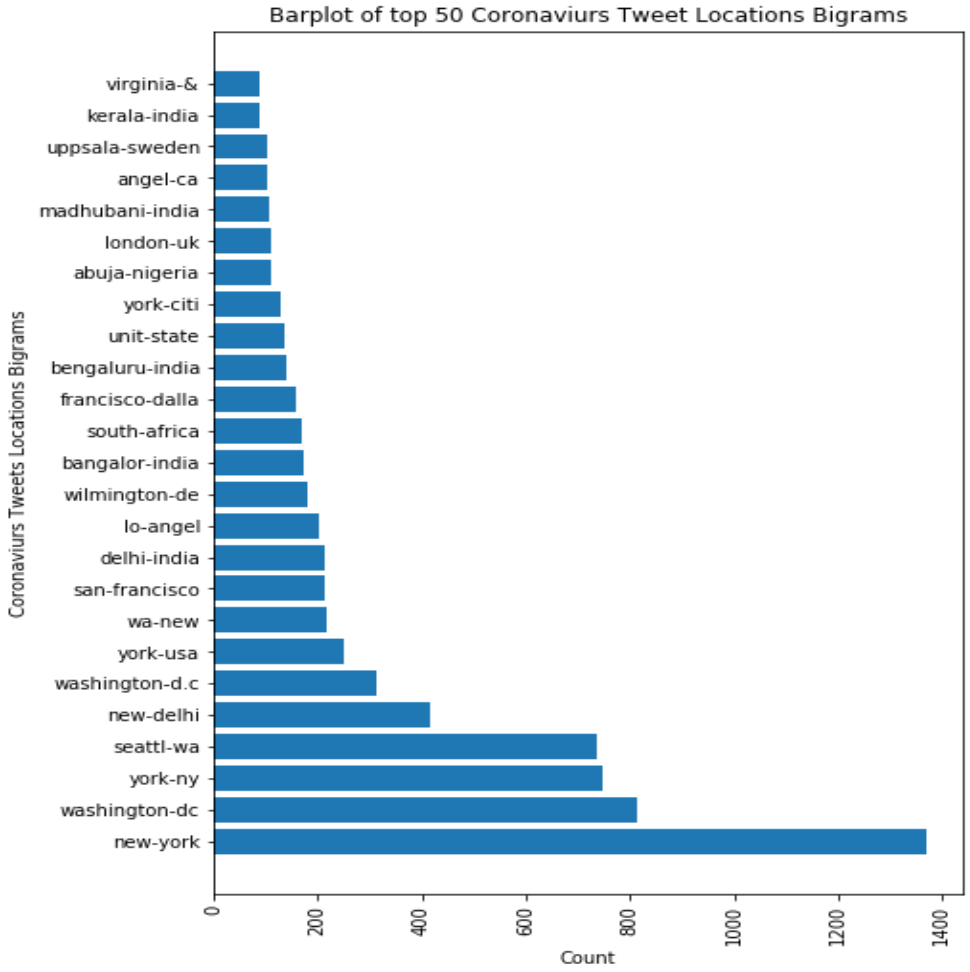
```
os.chdir('B:\\Travel\\US\\USF_Course\\TextAnalytics\\TextAnalyticsProject\\Co
ronavirusReasonsForRetweet\\')

df_a = pd.read_csv('tweets_28_3_2020_16_25.csv')
df_b = pd.read_csv('tweets_29_3_2020_16_59.csv')
df_c = pd.read_csv('tweets_29_3_2020_17_3.csv')
df_d = pd.read_csv('tweets_30_3_2020_21_35.csv')
df_e = pd.read_csv('tweets_30_3_2020_22_9.csv')
df_f= pd.read_csv('tweets_30_3_2020_22_35.csv')
df_g=pd.read_csv('tweets_5_4_2020_13_42.csv')
df_h=pd.read_csv('tweets_9_4_2020_15_22.csv')
df_i=pd.read_csv('tweets_14_4_2020_21_57.csv')
df_j=pd.read_csv('tweets_19_4_2020_0_16.csv')
df_k=pd.read_csv('tweets_22_4_2020_15_27.csv')
df = df_a.append([df_b, df_c, df_d, df_e, df_f, df_g, df_h, df_i, df_j,
df_k])
df = df.fillna('blank')
df=df.reset_index()
col_name = 'full_text'
output_col_name_1 = 'cleaned_text'
output_col_name_2 = 'cleaned_text_no_hashtag'
df = data_preprocessing(df, col_name, output_col_name_1,output_col_name_2 )
col_name = 'cleaned_text_no_hashtag'
output_col_name_1 = 'tokens'
output_col_name_2 = 'tokens_bigrams'
df = create_tokens_stem_lem_bigrams(df, col_name, output_col_name_1,
output_col_name_2)
df.loc[1, 'cleaned_text_no_hashtag']
u'this really gross shouldnt surpriseddemocrats have been pushing conspiracy
theories with abandon over years barbara boxer baselessly accuses president
trump covering wuhanvirus pushback from'
bigrams_freq = create_bigram_frequency(df['tokens_bigrams'])
bigrams_freq.sort(reverse=True)
top25_bigrams = bigrams_freq[0:25]
values, bigrams = zip(*top25_bigrams)
bigrams_joined = [w[0]+"-"+w[1] for w in bigrams]
df.shape
(15829, 16)

fig = plt.figure(figsize=(7,10))
plt.barh(bigrams_joined, values)
plt.ylabel('Coronaviurs Tweets Bigrams')
plt.xlabel('Count')
plt.xticks(rotation=90)
plt.title('Barplot of top 50 Coronaviurs Tweet Bigrams')
plt.show()
```



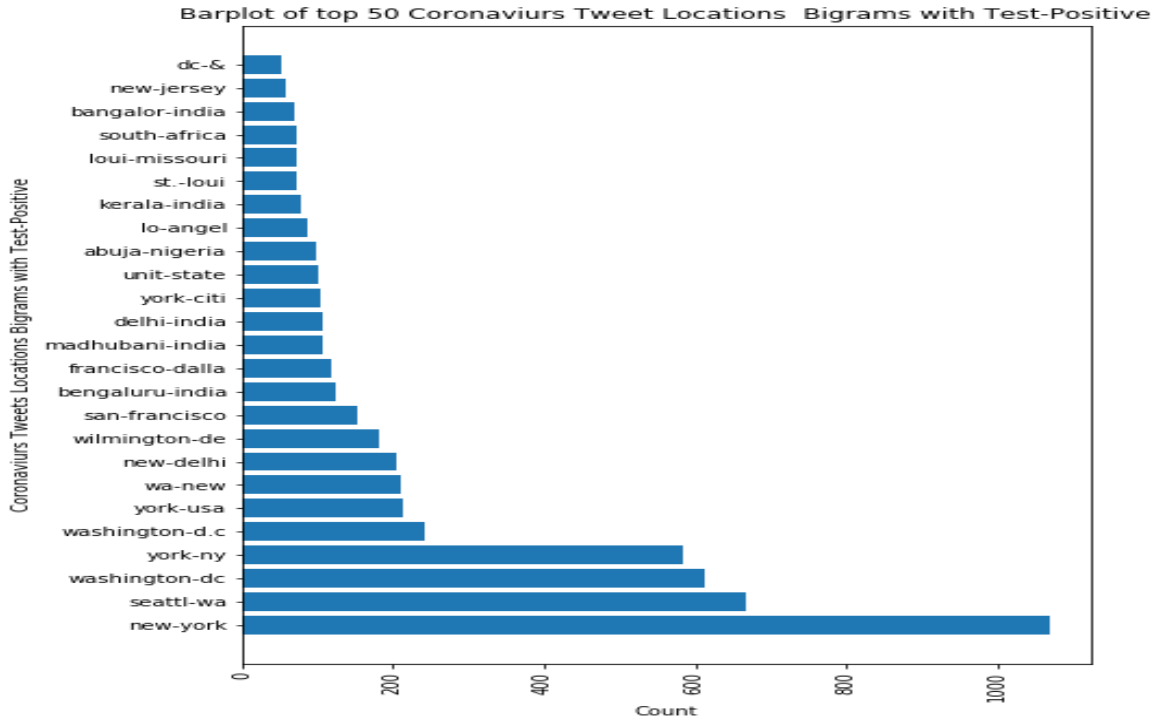
Predict the locations with highest incidents of coronavirus

```
col_name = 'location'
output_col_name_1 = 'cleaned_location'
output_col_name_2 = 'cleaned_location_1'
df = data_preprocessing(df, col_name, output_col_name_1, output_col_name_2)
df['cleaned_location'] = df['cleaned_location'].str.replace('
', 'blank')
#df['cleaned_location'] =
df['cleaned_location'].replace(regex=['^.*india.*$'], value='bharath')
```

```
df['cleaned_location'] = df['cleaned_location'].replace(regex=['^.*az.*$'],
value='arizona')

df['created_at_datetime'] = pd.to_datetime(df['created_at'], format='%Y-%m-%d
%H:%M:%S')
df['created_at_date'] = df['created_at_datetime'].dt.date
col_name = 'cleaned_location'
output_col_name_1 = 'tokens_location'
output_col_name_2 = 'tokens_location_bigram'
df = create_tokens_stem_lem_bigrams(df, col_name, output_col_name_1,
output_col_name_2)
df = df.drop(['index', 'Unnamed: 0'], axis=1)
bigrams_freq_location = create_bigram_frequency(df['tokens_location_bigram'])
bigrams_freq_location.sort(reverse=True)
top25_bigrams_location = bigrams_freq_location[0:25]
values_location, bigrams_location = zip(*top25_bigrams_location)
bigrams_joined_location = [w[0]+"-"+w[1] for w in bigrams_location]
fig = plt.figure(figsize=(7,10))
plt.barh(bigrams_joined_location, values_location)
plt.ylabel('Coronaviurs Tweets Locations Bigrams')
plt.xlabel('Count')
plt.xticks(rotation=90)
plt.title('Barplot of top 50 Coronaviurs Tweet Locations Bigrams')
plt.show()
```



```
bigrams = [('test', 'posit'), ('confirm', 'case'), ('uwmedicin', 'posit'), \
          ('covid', 'patient'), ('posit', 'case'), ('posit', 'covid'),
('coronaviru', 'case')]

#words = ['test', 'posit', 'case']
words = ['test', 'posit', 'confirm', 'case', 'uwmedicin', 'posit', 'covid',
'patient',\
            'posit', 'case', 'posit', 'covid', 'coronaviru', 'case']
pattern = '|'.join(words)
input_col = 'tokens'
output_col_1 = 'token_string'
output_col_2 = 'test_positive_y_n'
df = count_occurrence_of_words(df, input_col, output_col_1, output_col_2,
pattern)
df_tmp = df[df['test_positive_y_n']==True]
bigrams_freq_location_positive =
create_bigram_frequency(df_tmp['tokens_location_bigram'])
bigrams_freq_location_positive.sort(reverse=True)
top25_bigrams_location_positive = bigrams_freq_location_positive[0:25]
values_location_positive, bigrams_location_positive =
zip(*top25_bigrams_location_positive)
bigrams_joined_location_positive = [w[0]+"-"+w[1] for w in
bigrams_location_positive]
fig = plt.figure(figsize=(7,10))
plt.barh(bigrams_joined_location_positive, values_location_positive)
plt.ylabel('Coronaviurs Tweets Locations Bigrams with Test-Positive')
plt.xlabel('Count')
plt.xticks(rotation=90)
plt.title('Barplot of top 50 Coronaviurs Tweet Locations  Bigrams with Test-
Positive')
plt.show()
```



```
df_loc = pd.read_csv('US_State_City.csv')
df_loc = df_loc.fillna('United States')
df_loc = df_loc.drop_duplicates()
df_loc.head()
```

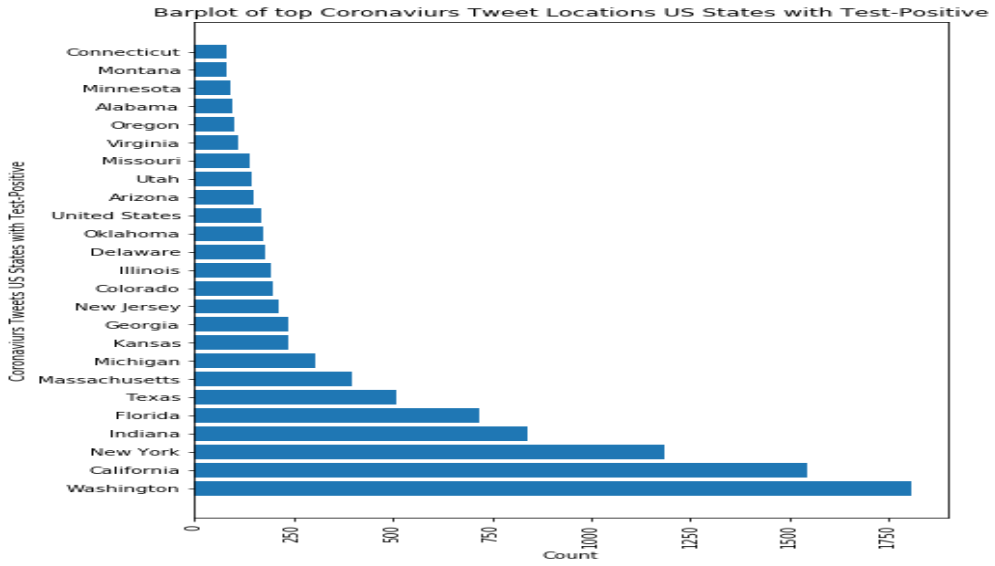
	Country	StateAbbr	StateDesc	CityName
0	US	CT	Connecticut	Hartford
1	US	NY	New York	New York
2	US	IL	Illinois	Chicago
3	US	TX	Texas	San Antonio
4	US	TX	Texas	Austin

```
df_loc_1 = pd.read_csv('world_cities.csv')
df_loc_1.columns=['CityName', 'Country', 'StateDesc']
list_of_countries = ['India', 'South Africa', 'United Kingdom', 'Japan',
'Indonesia', 'Phillipines', \
                     'South Korea', 'Brazil', 'Mexico', 'Bangladesh', 'Egypt',
'Pakistan', \
                     'Russia', 'Thailand', 'Argentina', 'Nigeria', 'Iran',
'Turkey', 'Congo', 'France', \
                     'Vietnam', 'Colombia', 'Taiwan', 'Malaysia', 'Iraq',
'Canada', 'Chile', \
                     'Spain', 'Saudi Arabia']
list_of_cities = ['Tokyo', 'Jakarta', 'Delhi', 'Manila', 'Seoul', 'Mumbai',
'Sao Paulo', \
                  'Mexico City', 'Dhaka', 'Osaka', 'Cairo', 'Karachi',
'Moscow', 'Bangkok', \
                 'Kolkata', 'Buenos Aires', 'Lagos', 'Tehran', 'Istanbul',
'Kinshasa', 'Rio de Janeiro', \
                 'Lahore', 'Lima', 'Bangalore', 'Paris', 'London', 'Bogota',
'Chennai', \
                 'Hyderabad', 'Johannesburg', 'Hanoi', 'Onitsha', 'Kuala
Lumpur', \
                 'Ahmedabad', 'Hong Kong', 'Baghdad', 'Dusseldorf',
'Toronto', \
                 'Santiago', 'Surat', 'Madrid', 'Pune', 'Riyadh']
df_loc_1 = df_loc_1.loc[df_loc_1['Country'].isin(list_of_countries)]
df_loc_1 = df_loc_1.loc[df_loc_1['CityName'].isin(list_of_cities)]
st = set(printable)
for column in df_loc_1.columns:
    df_loc_1[column] = df_loc_1[column].apply(lambda x: ''.join([" " if i not
in st else i for i in x]))
df_loc_1.head()
```

	CityName	Country	StateDesc
188	Buenos Aires	Argentina	Buenos Aires F.D.
699	Dhaka	Bangladesh	Dhaka
1560	Santiago	Brazil	Rio Grande do Sul
1602	Rio de Janeiro	Brazil	Rio de Janeiro

	CityName	Country	StateDesc
2465	London	Canada	Ontario

```
df_list = [df_loc[['CityName', 'Country', 'StateDesc']],df_loc_1[['CityName',
'Country', 'StateDesc']]]
df_loc_master = pd.concat(df_list, ignore_index=True)
os.chdir('B:\Travel\US\USF_Course\TextAnalytics\TextAnalyticsProject\Coronavi
rusReasonsForRetweet')
df_loc_master.to_csv('location_master.csv')
data_location_col= 'cleaned_location'
loc_master_state_col='StateDesc'
loc_master_city_col='CityName'
loc_master_country_col='Country'
output_state_col='cleaned_location_state'
output_country_col='cleaned_location_country'

df = standardize_location(df, df_loc_master, data_location_col,
loc_master_state_col, loc_master_city_col, \
                          loc_master_country_col, output_state_col,
output_country_col)
df_tmp_1 = df[df['test_positive_y_n']==True]
df_tmp_1 = df[df['cleaned_location_country']=='US']
#df_tmp_1 = df[df['cleaned_location_state']!='Unknown']
location_freq_positive =
create_location_frequency(df_tmp_1['cleaned_location_state'])
location_freq_positive.sort(reverse=True)
top25_location_positive = location_freq_positive[0:25]
values_location_positive_1, location_positive_1 =
zip(*top25_location_positive)
fig = plt.figure(figsize=(7,10))
plt.barh(location_positive_1, values_location_positive_1)
plt.ylabel('Coronaviurs Tweets US States with Test-Positive')
plt.xlabel('Count')
plt.xticks(rotation=90)
plt.title('Barplot of top Coronaviurs Tweet Locations US States with Test-
Positive')
plt.show()
```



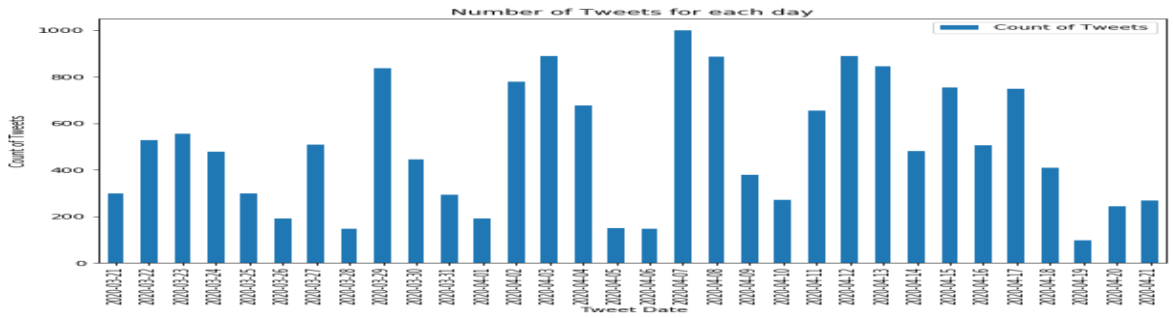
```
os.chdir("B:\Travel\US\USF_Course\TextAnalytics\TextAnalyticsProject\Coronavi
rusReasonsForRetweet")
df.to_csv('coronavirus_tweets_cleaned_1.csv')
df_processed = pd.read_csv('coronavirus_tweets_cleaned.csv')
df_select =
df_processed[['created_at_date','cleaned_location_country','test_positive_y_n
', 'cleaned_location_state']]
df_select = df_select[df_select['cleaned_location_country'] == 'US']
df_select = df_select[df_select['test_positive_y_n']==True]
df_select = df_select[['cleaned_location_state', 'created_at_date',
'cleaned_location_country']]
df_select_grouped =
df_select.groupby(['cleaned_location_state','created_at_date']).count().reset
_index()
df_select_grouped.to_csv('coronavirus_tweets_cleaned_grouped.csv')
words = ['die', 'death']
pattern = '|'.join(words)
input_col = 'tokens'
output_col_1 = 'token_string'
output_col_2 = 'death_y_n'
df = count_occurrence_of_words(df, input_col, output_col_1, output_col_2,
pattern)
```

Descriptive Analysis

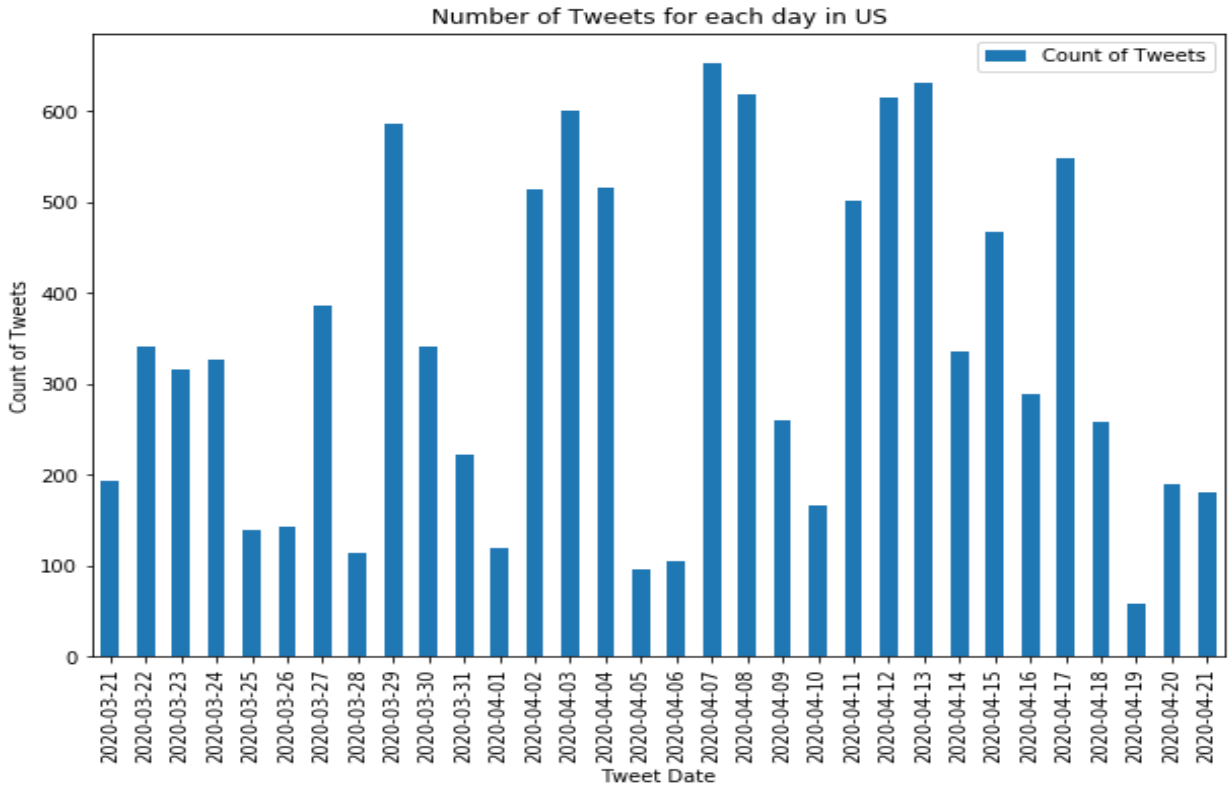
```
df_bar = df_processed[['created_at_date', 'cleaned_location_country']]

df_bar_grouped=df_bar.groupby(['created_at_date']).count().reset_index()
df_bar_grouped.columns=['Tweet Date', 'Count of Tweets']
df_bar_grouped.head()
```

	Tweet Date	Count of Tweets
0	2020-03-21	299
1	2020-03-22	528
2	2020-03-23	555
3	2020-03-24	477
4	2020-03-25	297

```
df_bar_grouped.plot(kind='bar', figsize=(10,7))
plt.xticks(np.arange(0,32,1), df_bar_grouped['Tweet Date'])
plt.xlabel('Tweet Date')
plt.ylabel('Count of Tweets')
plt.title('Number of Tweets for each day')
plt.show()
```

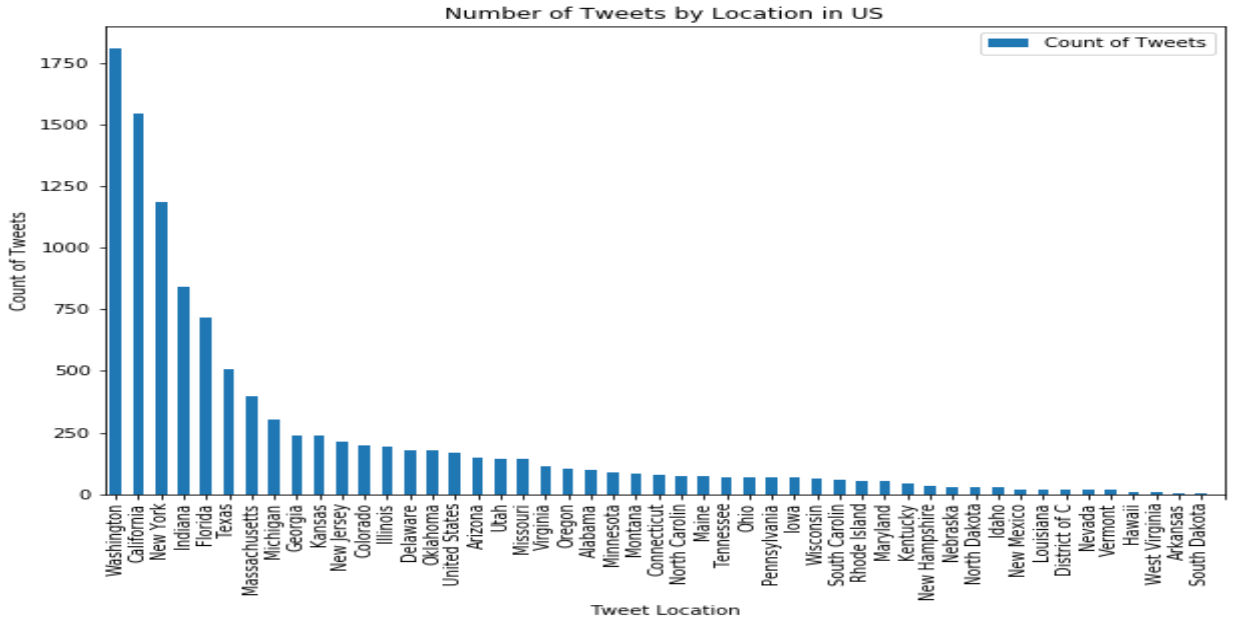



```
df_bar_1 = df_processed[['created_at_date', 'cleaned_location_country']]
df_bar_1 = df_bar[df_bar['cleaned_location_country']=='US']
df_bar_grouped_1=df_bar_1.groupby(['created_at_date']).count().reset_index()
df_bar_grouped_1.columns=['Tweet Date', 'Count of Tweets']
df_bar_grouped_1.plot(kind='bar', figsize=(10,7))
plt.xticks(np.arange(0,32,1), df_bar_grouped_1['Tweet Date'])
plt.xlabel('Tweet Date')
plt.ylabel('Count of Tweets')
plt.title('Number of Tweets for each day in US')
plt.show()
```




```
df_bar_2 = df_processed[df_processed['cleaned_location_country']=='US']
df_bar_2 = df_bar_2[['created_at_date', 'cleaned_location_state']]
df_bar_grouped_2 =
df_bar_2.groupby('cleaned_location_state').count().reset_index()
df_bar_grouped_2.columns = ['Tweet Location', 'Count of Tweets']
df_bar_grouped_2 = df_bar_grouped_2.sort_values('Count of Tweets',
ascending=False)
```

```
df_bar_grouped_2.plot(kind='bar', figsize=(10,7))
plt.xticks(np.arange(0,50,1), df_bar_grouped_2['Tweet Location'])
plt.xlabel('Tweet Location')
plt.ylabel('Count of Tweets')
plt.title('Number of Tweets by Location in US')
plt.show()
```




```
df_bar_grouped_2.groupby('Tweet Location').sum().sum()
Count of Tweets    10820
dtype: int64
```

John Hopkins

```
df_jh = pd.read_csv('john-hopkins-date-column.csv')
df_jh.head()
```

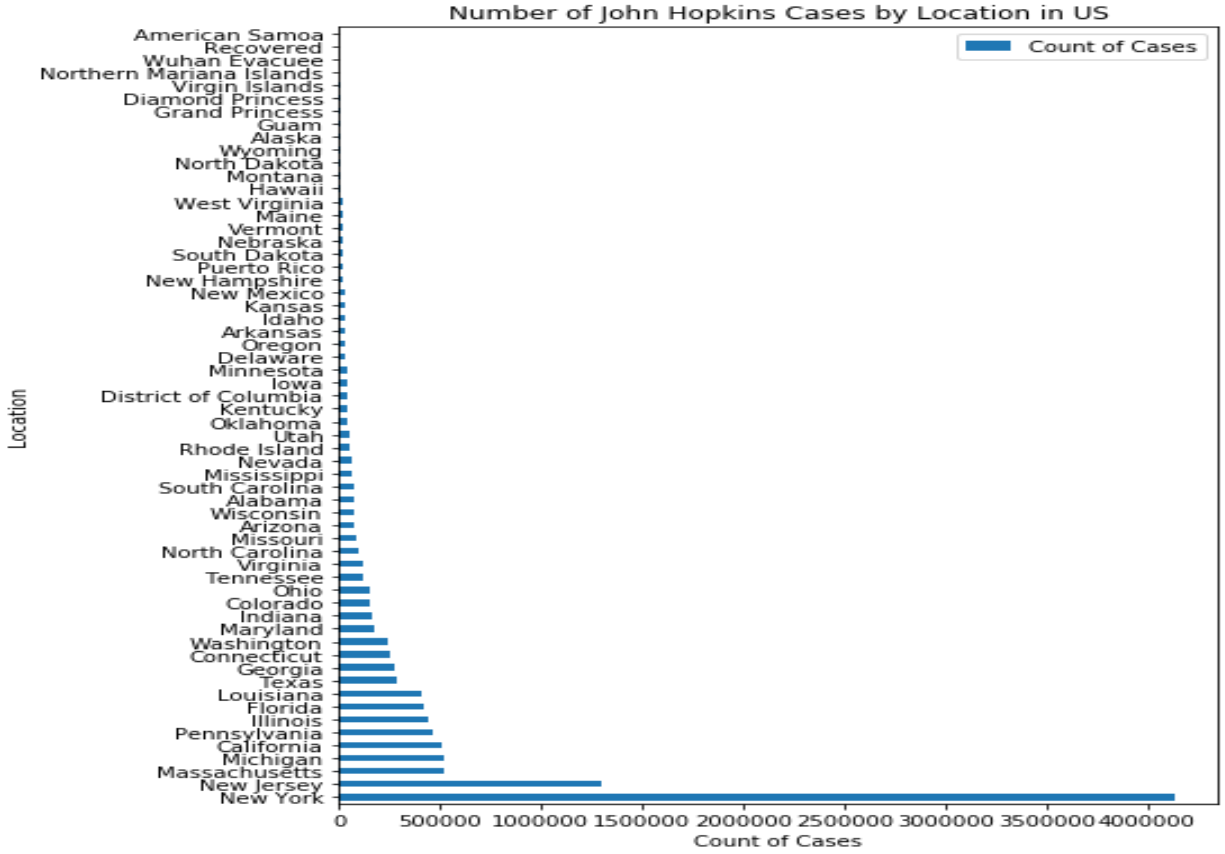
	Unnamed: 0	Province_State	Country_Region	Last_Update	Confirmed	Deaths	date
0	0	NaN	NaN	NaN	67800	3139	NaN
1	1	NaN	NaN	NaN	53578	4825	NaN
2	2	NaN	NaN	NaN	25374	1375	NaN
3	3	NaN	NaN	NaN	22213	84	NaN
4	4	NaN	NaN	NaN	20610	1556	NaN

```
df_jh = df_jh[df_jh['Country_Region']=='US']
df_jh.head()
```

	Unnamed: 0	Province_State	Country_Region	Last_Update	Confirmed	Deaths	date
304	304	New York	US	2020-03-22 23:45:00	9654	63	2020-03-22

	Unnamed: 0	Province_State	Country_Region	Last_Update	Confirmed	Deaths	date
305	305	New York	US	2020-03-22 23:45:00	1900	4	2020-03-22
306	306	New York	US	2020-03-22 23:45:00	1873	0	2020-03-22
307	307	New York	US	2020-03-22 23:45:00	1034	9	2020-03-22
308	308	New York	US	2020-03-22 23:45:00	455	1	2020-03-22

```
df_jh_grouped=df_jh[['Province_State', 'Confirmed']]
df_jh_grouped = df_jh_grouped.groupby(['Province_State']).sum().reset_index()
df_jh_grouped.columns = ['Location', 'Count of Cases']
df_jh_grouped = df_jh_grouped.sort_values('Count of Cases', ascending=False)
df_jh_grouped.plot(kind='barh', figsize=(7,10))
plt.yticks(np.arange(0,60,1), df_jh_grouped['Location'])
plt.ylabel('Location')
plt.xlabel('Count of Cases')
plt.title('Number of John Hopkins Cases by Location in US')
plt.show()
```




```
df_jh_ny = df_jh[df_jh['Province_State']=='New York']
df_jh_ny = df_jh_ny[['date', 'Province_State', 'Confirmed']]
```

```
df_jh_ny = df_jh_ny.groupby(['date', 'Province_State']).sum().reset_index()
df_jh_ny.columns=['created_at_date', 'cleaned_location_state', 'cases_count']
df_jh_ny.head()
```

	created_at_date	cleaned_location_state	cases_count
0	2020-03-22	New York	15800
1	2020-03-23	New York	20884
2	2020-03-24	New York	25681
3	2020-03-25	New York	30841
4	2020-03-26	New York	37877

```
df_ny=df_processed[df_processed['cleaned_location_state']=='New York']
df_ny=df_ny[df_ny['test_positive_y_n']==True]
df_ny = df_ny[['created_at_date', 'cleaned_location_state',
'test_positive_y_n']]
df_ny_grouped =
df_ny.groupby(['created_at_date','cleaned_location_state']).count().reset_ind
ex()
df_ny_grouped.columns=['created_at_date', 'cleaned_location_state',
'cases_count']

df_ny=df_processed[df_processed['cleaned_location_state']=='New York']
df_ny=df_ny[df_ny['test_positive_y_n']==True]
df_ny = df_ny[['created_at_date', 'cleaned_location_state',
'test_positive_y_n']]
df_ny_grouped =
df_ny.groupby(['created_at_date','cleaned_location_state']).count().reset_ind
ex()
df_ny_grouped.columns=['created_at_date', 'cleaned_location_state',
'cases_count']
df_ny_grouped.head()

df_ny_grouped.head()
```

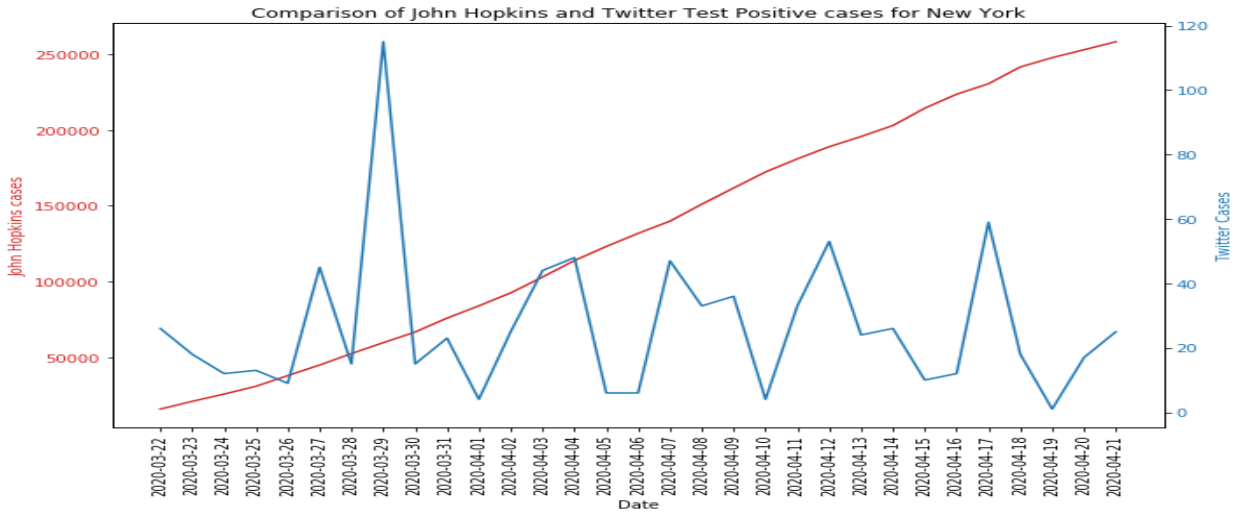
	created_at_date	cleaned_location_state	cases_count
0	2020-03-21	New York	18
1	2020-03-22	New York	26
2	2020-03-23	New York	18
3	2020-03-24	New York	12
4	2020-03-25	New York	13

```
df_ny_grouped_merged = df_jh_ny.merge(df_ny_grouped, on=['created_at_date',
'cleaned_location_state'])
df_ny_grouped_merged.head()
```

	created_at_date	cleaned_location_state	cases_count_x	cases_count_y
0	2020-03-22	New York	15800	26

	created_at_date	cleaned_location_state	cases_count_x	cases_count_y
1	2020-03-23	New York	20884	18
2	2020-03-24	New York	25681	12
3	2020-03-25	New York	30841	13
4	2020-03-26	New York	37877	9

```
t = df_ny_grouped_merged['created_at_date']
data1 = df_ny_grouped_merged['cases_count_x']
data2 = df_ny_grouped_merged['cases_count_y']

fig, ax1 = plt.subplots(figsize=(10,7))

color = 'tab:red'
ax1.set_xlabel('Date')
ax1.set_ylabel('John Hopkins cases', color=color)
ax1.plot(t, data1, color=color)
ax1.tick_params(axis='y', labelcolor=color)
ax1.tick_params(axis='x', rotation=90)
ax1.set_title('Comparison of John Hopkins and Twitter Test Positive cases for
New York')

ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

color = 'tab:blue'
ax2.set_ylabel('Twitter Cases', color=color)  # we already handled the x-
label with ax1
ax2.plot(t, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```



California

```
df_jh_ca = df_jh[df_jh['Province_State']=='California']
```

```
df_jh_ca = df_jh_ca[['date', 'Province_State', 'Confirmed']]
df_jh_ca = df_jh_ca.groupby(['date', 'Province_State']).sum().reset_index()
df_jh_ca.columns=['created_at_date', 'cleaned_location_state', 'cases_count']
df_jh_ca.head()
```

	created_at_date	cleaned_location_state	cases_count
0	2020-03-22	California	1646
1	2020-03-23	California	2108
2	2020-03-24	California	2538
3	2020-03-25	California	2998
4	2020-03-26	California	3899

```
df_ca=df_processed[df_processed['cleaned_location_state']=='California']
df_ca=df_ca[df_ca['test_positive_y_n']==True]
df_ca = df_ca[['created_at_date', 'cleaned_location_state',
'test_positive_y_n']]
df_ca_grouped =
df_ca.groupby(['created_at_date','cleaned_location_state']).count().reset_ind
ex()
df_ca_grouped.columns=['created_at_date', 'cleaned_location_state',
'cases_count']
df_ca_grouped.head()
```

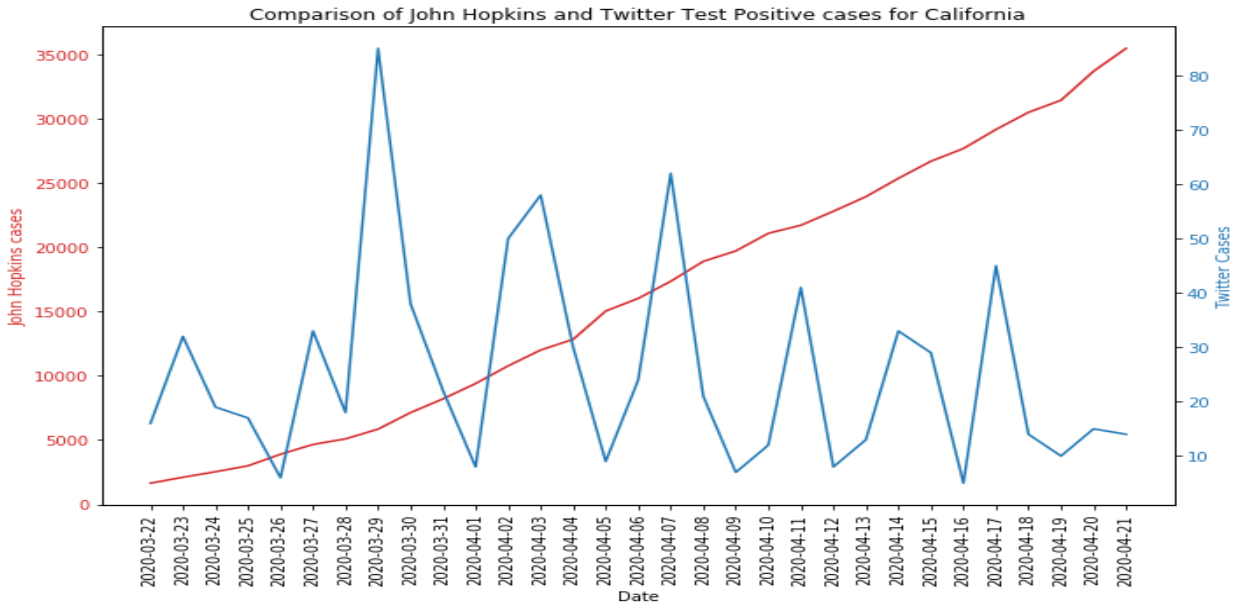
	created_at_date	cleaned_location_state	cases_count
0	2020-03-21	California	15
1	2020-03-22	California	16
2	2020-03-23	California	32
3	2020-03-24	California	19
4	2020-03-25	California	17

```
df_ca_grouped_merged = df_jh_ca.merge(df_ca_grouped, on=['created_at_date',
'cleaned_location_state'])
df_ca_grouped_merged.head()
```

	created_at_date	cleaned_location_state	cases_count_x	cases_count_y
0	2020-03-22	California	1646	16
1	2020-03-23	California	2108	32
2	2020-03-24	California	2538	19
3	2020-03-25	California	2998	17
4	2020-03-26	California	3899	6

```
t = df_ca_grouped_merged['created_at_date']
data1 = df_ca_grouped_merged['cases_count_x']
data2 = df_ca_grouped_merged['cases_count_y']
```

```
fig, ax1 = plt.subplots(figsize=(10,7))

color = 'tab:red'
ax1.set_xlabel('Date')
ax1.set_ylabel('John Hopkins cases', color=color)
ax1.plot(t, data1, color=color)
ax1.tick_params(axis='y', labelcolor=color)
ax1.tick_params(axis='x', rotation=90)
ax1.set_title('Comparison of John Hopkins and Twitter Test Positive cases for
California')

ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

color = 'tab:blue'
ax2.set_ylabel('Twitter Cases', color=color)  # we already handled the x-
label with ax1
ax2.plot(t, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```



Washington

```
df_jh_wa = df_jh[df_jh['Province_State']=='Washington']
df_jh_wa = df_jh_wa[['date', 'Province_State', 'Confirmed']]
df_jh_wa = df_jh_wa.groupby(['date', 'Province_State']).sum().reset_index()
df_jh_wa.columns=['created_at_date', 'cleaned_location_state', 'cases_count']
df_jh_wa.head()
```

	created_at_date	cleaned_location_state	cases_count
0	2020-03-22	Washington	1997
1	2020-03-23	Washington	2221

	created_at_date	cleaned_location_state	cases_count
2	2020-03-24	Washington	2328
3	2020-03-25	Washington	2591
4	2020-03-26	Washington	3207

```
df_wa=df_processed[df_processed['cleaned_location_state']=='Washington']
df_wa=df_wa[df_wa['test_positive_y_n']==True]
df_wa = df_wa[['created_at_date', 'cleaned_location_state',
'test_positive_y_n']]
df_wa_grouped =
df_wa.groupby(['created_at_date','cleaned_location_state']).count().reset_ind
ex()
df_wa_grouped.columns=['created_at_date', 'cleaned_location_state',
'cases_count']
df_wa_grouped.head()
```

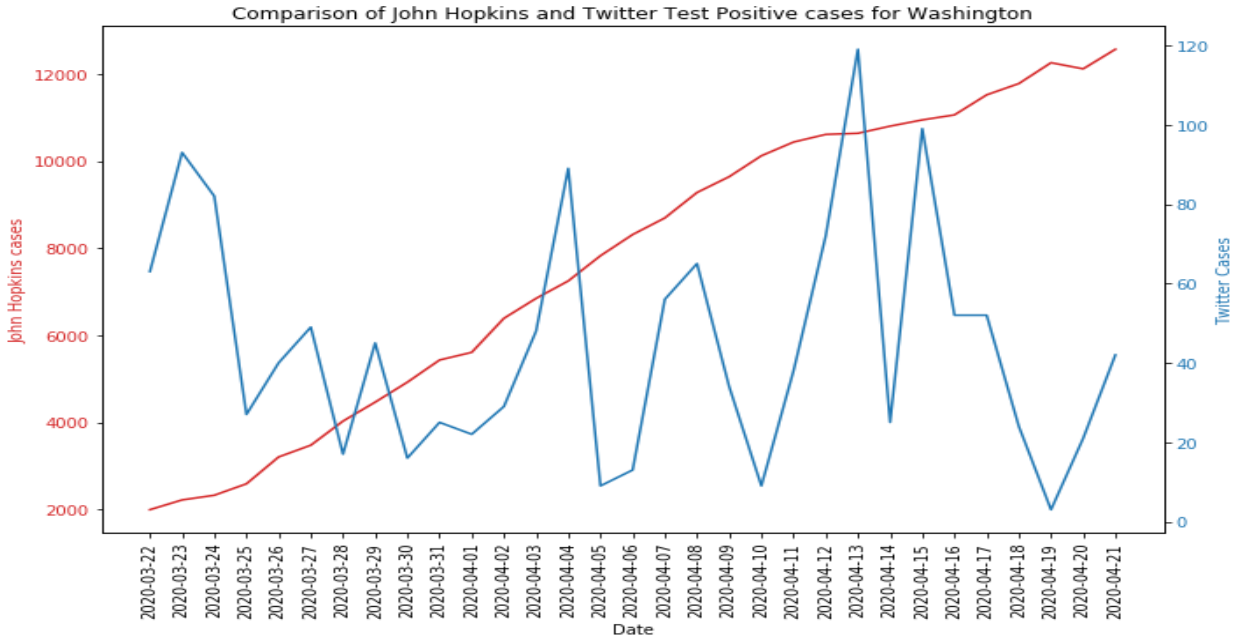
	created_at_date	cleaned_location_state	cases_count
0	2020-03-21	Washington	20
1	2020-03-22	Washington	63
2	2020-03-23	Washington	93
3	2020-03-24	Washington	82
4	2020-03-25	Washington	27

```
df_wa_grouped_merged = df_jh_wa.merge(df_wa_grouped, on=['created_at_date',
'cleaned_location_state'])
df_wa_grouped_merged.head()
```

	created_at_date	cleaned_location_state	cases_count_x	cases_count_y
0	2020-03-22	Washington	1997	63
1	2020-03-23	Washington	2221	93
2	2020-03-24	Washington	2328	82
3	2020-03-25	Washington	2591	27
4	2020-03-26	Washington	3207	40

```
t = df_wa_grouped_merged['created_at_date']
data1 = df_wa_grouped_merged['cases_count_x']
data2 = df_wa_grouped_merged['cases_count_y']

fig, ax1 = plt.subplots(figsize=(10,7))

color = 'tab:red'
ax1.set_xlabel('Date')
ax1.set_ylabel('John Hopkins cases', color=color)
ax1.plot(t, data1, color=color)
ax1.tick_params(axis='y', labelcolor=color)
ax1.tick_params(axis='x', rotation=90)
```

```
ax1.set_title('Comparison of John Hopkins and Twitter Test Positive cases for
Washington')

ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

color = 'tab:blue'
ax2.set_ylabel('Twitter Cases', color=color)  # we already handled the x-
label with ax1
ax2.plot(t, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```



Indiana

```
df_jh_in = df_jh[df_jh['Province_State']=='Indiana']
df_jh_in = df_jh_in[['date', 'Province_State', 'Confirmed']]
df_jh_in= df_jh_in.groupby(['date', 'Province_State']).sum().reset_index()
df_jh_in.columns=['created_at_date', 'cleaned_location_state', 'cases_count']
df_jh_in.head()
```

	created_at_date	cleaned_location_state	cases_count
0	2020-03-22	Indiana	204
1	2020-03-23	Indiana	270
2	2020-03-24	Indiana	368
3	2020-03-25	Indiana	477
4	2020-03-26	Indiana	645

```
df_in=df_processed[df_processed['cleaned_location_state']=='Indiana']
df_in=df_in[df_in['test_positive_y_n']==True]
```

```
df_in = df_in[['created_at_date', 'cleaned_location_state',
'test_positive_y_n']]
df_in_grouped =
df_in.groupby(['created_at_date','cleaned_location_state']).count().reset_ind
ex()
df_in_grouped.columns=['created_at_date', 'cleaned_location_state',
'cases_count']
df_in_grouped.head()
```

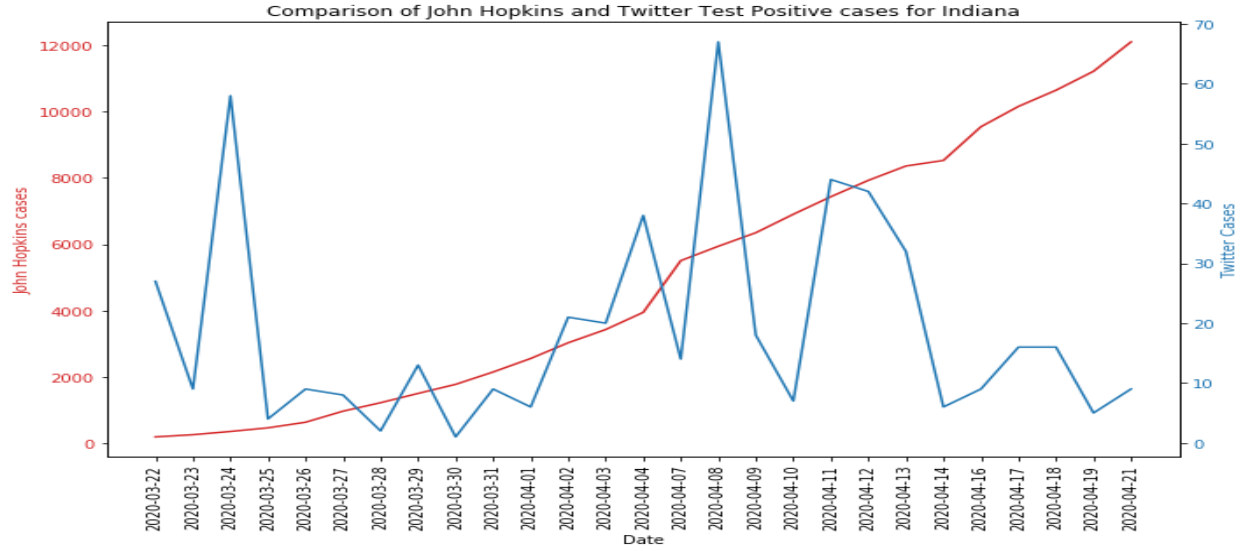
	created_at_date	cleaned_location_state	cases_count
0	2020-03-21	Indiana	9
1	2020-03-22	Indiana	27
2	2020-03-23	Indiana	9
3	2020-03-24	Indiana	58
4	2020-03-25	Indiana	4

```
df_in_grouped_merged = df_jh_in.merge(df_in_grouped, on=['created_at_date',
'cleaned_location_state'])
df_in_grouped_merged.head()
```

	created_at_date	cleaned_location_state	cases_count_x	cases_count_y
0	2020-03-22	Indiana	204	27
1	2020-03-23	Indiana	270	9
2	2020-03-24	Indiana	368	58
3	2020-03-25	Indiana	477	4
4	2020-03-26	Indiana	645	9

```
t = df_in_grouped_merged['created_at_date']
data1 = df_in_grouped_merged['cases_count_x']
data2 = df_in_grouped_merged['cases_count_y']

fig, ax1 = plt.subplots(figsize=(10,7))

color = 'tab:red'
ax1.set_xlabel('Date')
ax1.set_ylabel('John Hopkins cases', color=color)
ax1.plot(t, data1, color=color)
ax1.tick_params(axis='y', labelcolor=color)
ax1.tick_params(axis='x', rotation=90)
ax1.set_title('Comparison of John Hopkins and Twitter Test Positive cases for
Indiana')

ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

color = 'tab:blue'
ax2.set_ylabel('Twitter Cases', color=color)  # we already handled the x-
label with ax1
ax2.plot(t, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)
```

```
fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```




```
df_jh.head()
```

	Unnamed: 0	Province_State	Country_Region	Last_Update	Confirmed	Deaths	date
304	304	New York	US	2020-03-22 23:45:00	9654	63	2020-03-22
305	305	New York	US	2020-03-22 23:45:00	1900	4	2020-03-22
306	306	New York	US	2020-03-22 23:45:00	1873	0	2020-03-22
307	307	New York	US	2020-03-22 23:45:00	1034	9	2020-03-22
308	308	New York	US	2020-03-22 23:45:00	455	1	2020-03-22

```
df.head()
```

5 rows × 25 columns

```
df_jh_de = df_jh[df_jh['Country_Region']=='US']
df_jh_de = df_jh[['date', 'Deaths']]
df_jh_de_grouped = df_jh_de.groupby('date').sum().reset_index()
df_jh_de_grouped.columns=['Date', 'Count of Deaths']
df_jh_de_grouped['Date'] = pd.to_datetime(df_jh_de_grouped['Date'])
df_jh_de_grouped.head()
```

	Date	Count of Deaths
0	2020-03-22	427

	Date	Count of Deaths
1	2020-03-23	552
2	2020-03-24	706
3	2020-03-25	942
4	2020-03-26	1209

```
df_de = df[df['cleaned_location_country']=='US']
df_de = df_de[['created_at_date', 'death_y_n']]
df_de = df_de[df_de['death_y_n']==True]
df_de_grouped = df_de.groupby('created_at_date').count().reset_index()
df_de_grouped.columns = ['Date', 'Count of Deaths']
df_de_grouped['Date'] = pd.to_datetime(df_de_grouped['Date'])
df_de_grouped.head()
```

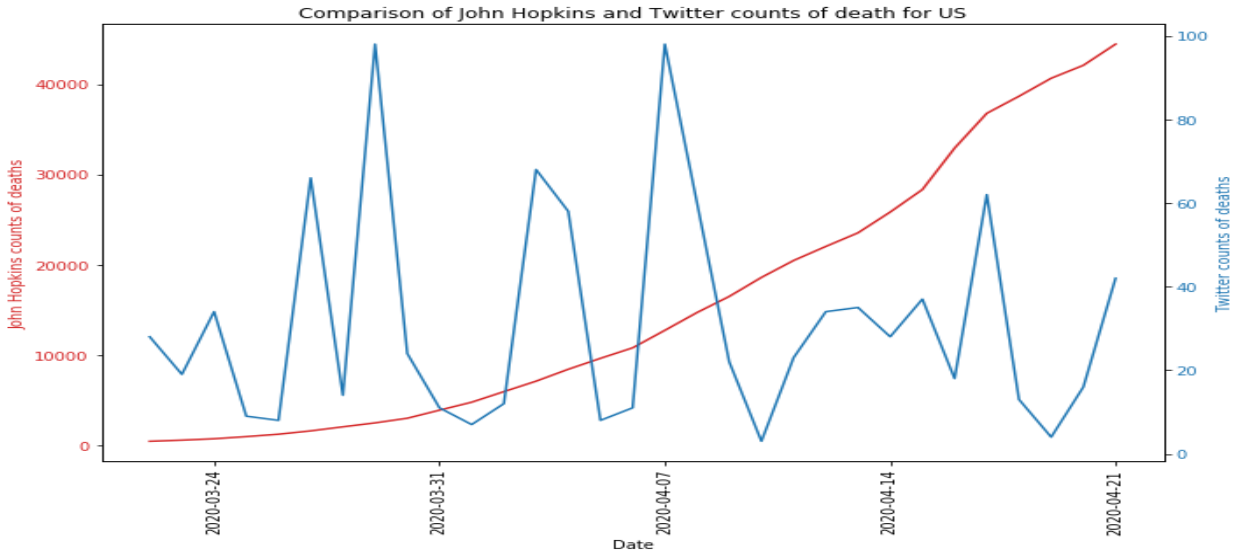
	Date	Count of Deaths
0	2020-03-21	18
1	2020-03-22	28
2	2020-03-23	19
3	2020-03-24	34
4	2020-03-25	9

```
df_de_grouped_merged = df_jh_de_grouped.merge(df_de_grouped, on=['Date'])
df_de_grouped_merged.head()
```

	Date	Count of Deaths_x	Count of Deaths_y
0	2020-03-22	427	28
1	2020-03-23	552	19
2	2020-03-24	706	34
3	2020-03-25	942	9
4	2020-03-26	1209	8

```
t = df_de_grouped_merged['Date']
data1 = df_de_grouped_merged['Count of Deaths_x']
data2 = df_de_grouped_merged['Count of Deaths_y']

fig, ax1 = plt.subplots(figsize=(10,7))

color = 'tab:red'
ax1.set_xlabel('Date')
ax1.set_ylabel('John Hopkins counts of deaths', color=color)
ax1.plot(t, data1, color=color)
ax1.tick_params(axis='y', labelcolor=color)
ax1.tick_params(axis='x', rotation=90)
ax1.set_title('Comparison of John Hopkins and Twitter counts of death for
US')
```

```
ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

color = 'tab:blue'
ax2.set_ylabel('Twitter counts of deaths', color=color)  # we already handled
the x-label with ax1
ax2.plot(t, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```




```
df_jh_cases = df_jh[df_jh['Country_Region']=='US']
df_jh_cases = df_jh[['date', 'Confirmed']]
df_jh_cases_grouped = df_jh_cases.groupby('date').sum().reset_index()
df_jh_cases_grouped.columns=['Date', 'Count of Cases']
df_jh_cases_grouped['Date'] = pd.to_datetime(df_jh_cases_grouped['Date'])
df_jh_cases_grouped.head()
```

	Date	Count of Cases
0	2020-03-22	33848
1	2020-03-23	43663
2	2020-03-24	53736
3	2020-03-25	65778
4	2020-03-26	83836

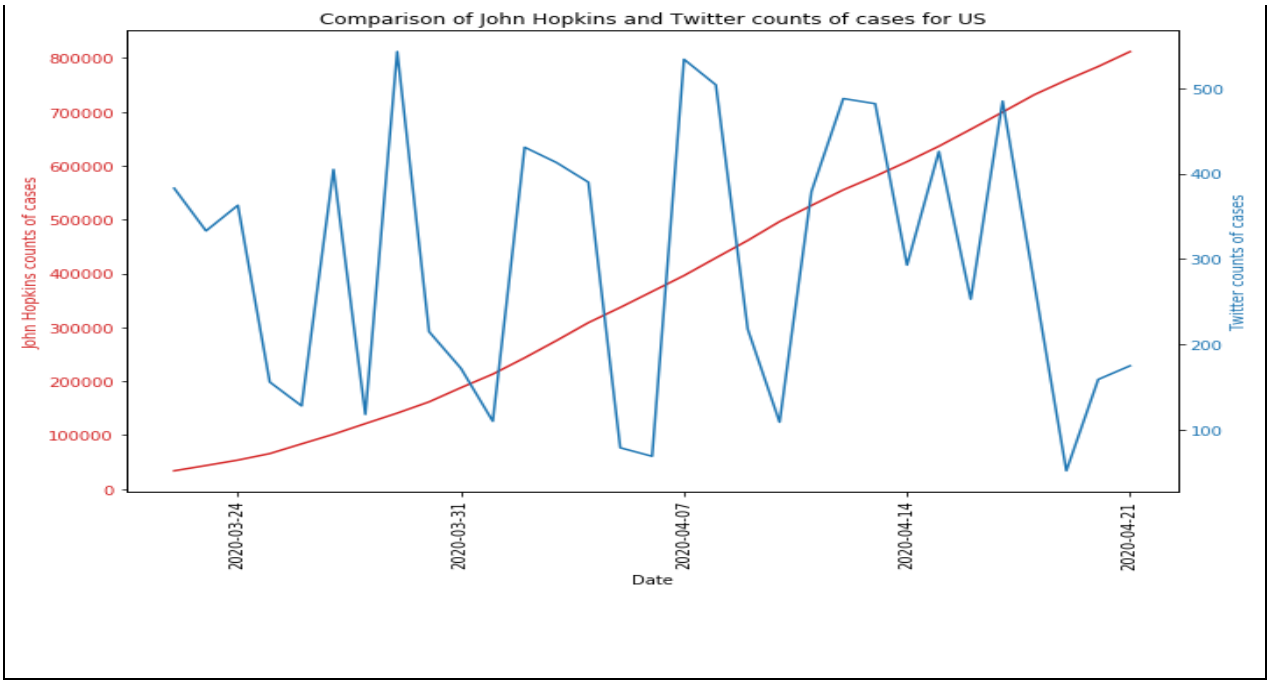
```
df_cases = df[df['cleaned_location_country']=='US']
df_cases = df[['created_at_date', 'test_positive_y_n']]
df_cases = df_cases[df_cases['test_positive_y_n']==True]
df_cases_grouped = df_cases.groupby('created_at_date').count().reset_index()
df_cases_grouped.columns = ['Date', 'Count of Cases']
df_cases_grouped['Date'] = pd.to_datetime(df_cases_grouped['Date'])
df_cases_grouped.head()
```

	Date	Count of Cases
0	2020-03-21	200
1	2020-03-22	383
2	2020-03-23	333
3	2020-03-24	363
4	2020-03-25	156

```
df_cases_grouped_merged = df_jh_cases_grouped.merge(df_cases_grouped,
on=['Date'])
df_cases_grouped_merged.head()
```

	Date	Count of Cases_x	Count of Cases_y
0	2020-03-22	33848	383
1	2020-03-23	43663	333
2	2020-03-24	53736	363
3	2020-03-25	65778	156
4	2020-03-26	83836	128

```
t = df_cases_grouped_merged['Date']
data1 = df_cases_grouped_merged['Count of Cases_x']
data2 = df_cases_grouped_merged['Count of Cases_y']

fig, ax1 = plt.subplots(figsize=(10,7))

color = 'tab:red'
ax1.set_xlabel('Date')
ax1.set_ylabel('John Hopkins counts of cases', color=color)
ax1.plot(t, data1, color=color)
ax1.tick_params(axis='y', labelcolor=color)
ax1.tick_params(axis='x', rotation=90)
ax1.set_title('Comparison of John Hopkins and Twitter counts of cases for
US')

ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

color = 'tab:blue'
ax2.set_ylabel('Twitter counts of cases', color=color)  # we already handled
the x-label with ax1
ax2.plot(t, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```


Comparison of John Hopkins and Twitter counts of cases for US
John Hopkins counts of cases
Twitter counts of cases
Date
0
100000
200000
300000
400000
500000
600000
700000
800000
100
200
300
400
500
2020-03-24
2020-03-31
2020-04-07
2020-04-14
2020-04-21