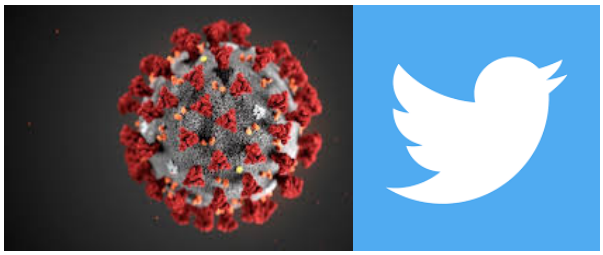
Covid-19's Impact on Twitter Trends



Team Members

Nick Zelada | Jacqueline Rivera

Md Rakibul Alam | Kehinde Alawode

Table of Contents

Introduction

Coronavirus (COVID19) was declared a global pandemic by the World Health Organization in March 2020. Since this declaration, many countries have enforced lockdowns and implemented social distancing as tactics to stop the spread of the virus. Lockdowns, social distancing, and quarantines have forced a new normal upon the lives of many people. With nothing to do and nowhere to go, social media usage has skyrocketed. Many platforms reported a huge increase in the number of daily users, including Twitter. Twitter is one of the most popular social media platforms with over 500 million tweets sent per day and has been popular for over a decade. In every tweet, users send their thoughts, sentiments, or perhaps share breaking news and Twitter stores these tweets in an archive. With the pandemic affecting everyone and with seemingly everyone using social media, can we measure the changes in the society by looking at twitter trends?

The goal of this project is to take a look at four COVID related trends and determine whether they translated into trends on Twitters. The four trends include impact on the elections while comparing it to 2016 elections, an increase in negative sentiment, Covid 19 health misinformation, and change in human mobility pattern. The Data set, motivation, experiments, results and conclusions for each trend is discussed in the sections that follow.

Data

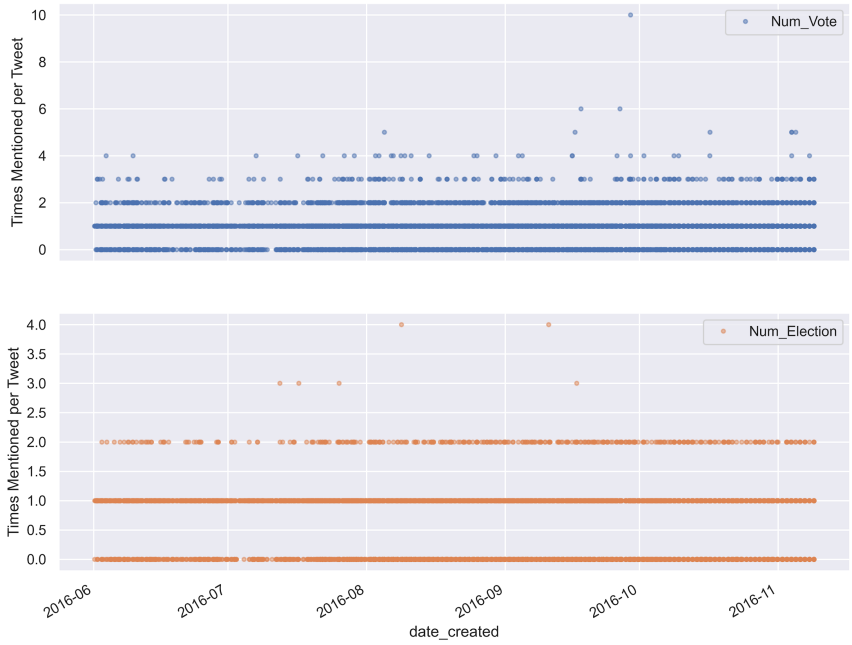
The data set used in this study are tweets originating from the United States (US). Most of the tweets are pulled from the (a) Twitter API and (b) Kaggle Tweet Dataset. A Twitter Developer Account was created using Twitter Apps (apps.twitter.com/) to retrieve data through Twitter Streaming API (Application Programming Interfaces). Python programming language was used to collect the data and associated Python libraries have been used. As the main focus of this study is geo-tagged tweets generated in the USA, the location-based data collection method has been used which has the ability to produce a more suitable and reliable dataset that serves the goal of the study. With Twitter's limit to retrieval of archived tweets, some tweets which formed part of the data set used in this study were also obtained from Kaggle.

Aim 1 - Covid 19's Impact on 2020 Election

2020 happens to be an election year, but an election like no other because of a pandemic. COVID has become a political subject on the campaign trail of the presidency and the media. In the curiosity of the question, if COVID had an impact on the 2020 election year while comparing it to the 2016 election year, time-series experiment and text mining were conducted. In the process, Twitter API was used to download data by filtering data using keywords, location, and date/time. For the 2016 dataset, 16,465 tweets from June 01, 2016, until election day of November 2016 were obtained while using the USA as the location and keyword "election" and "vote." As to the 2020 data, 16,465 tweets were downloaded with dates being June 01, 2020, until election day of 2020, while using the same location and keywords. Downloading the same number of tweets for each year allowed for a constant number when comparing them and the constant keywords since they generate election talk.

In 2016, the keyword "election" users said it 10,478 times, while having the keyword "vote" mentioned 12,667. For the 2020 dataset, the keyword "election" users said it 13,060 times while "vote" was 19,253.

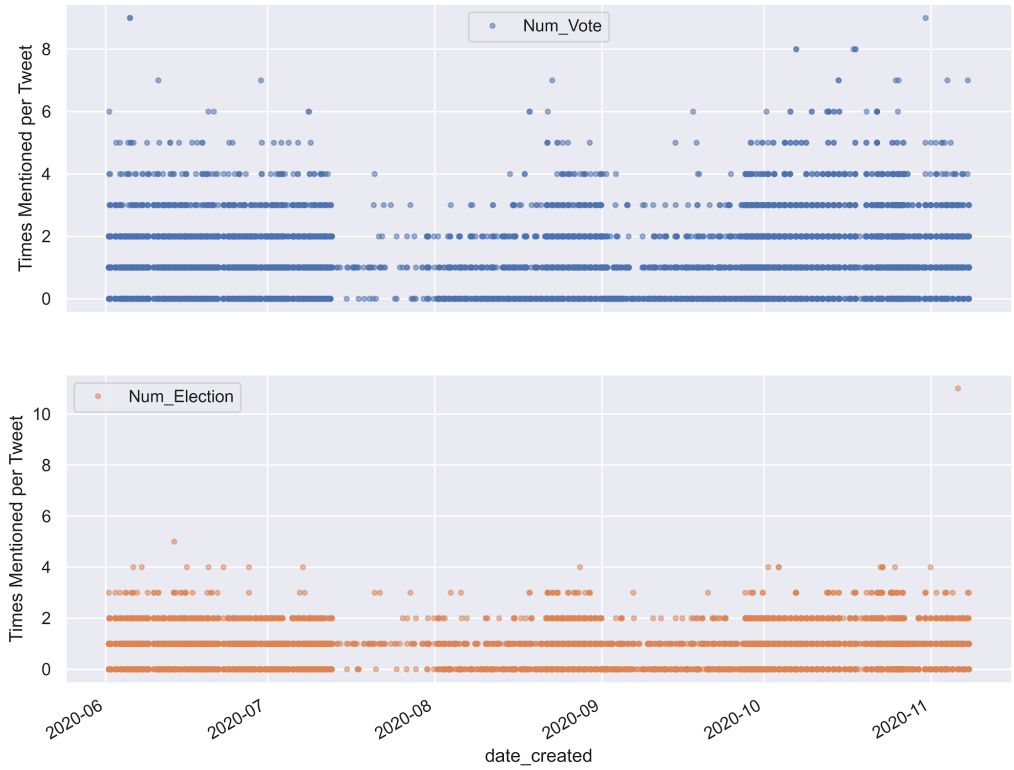
A creation of time series for the 2016 data created graphs for the keywords obtained. At first, we have the keyword "Vote" throughout the dataset, the number of times being zero, one, and two times staying pretty much constant through the tweets. Meaning some tweets did not have many mentions of "vote" to others having one to two times in the tweet. From time to time, there were three times to even four. The election was slightly different; through the dataset, the number of times mentioned was pretty substantial for one time per tweet to even zero times; here and there, it was two times.



For the 2020 dataset, creating a time series allowed data visualization of the plots for both keywords, noticing that they were a bit fuller. When it came to the number of times "vote" was mentioned during a tweet seem to be increased. For June, it was up to five times per tweet

throughout the month; it was pretty definite that tweets had mentions being from zero to five times per tweet. For August, it came down a bit, but then it increased for September. As time got closer to election day, the numbers mentioned per tweet increased to November, having some tweets reaching eight mentions of "vote" in a tweet.

When it came to the keyword "election," there was a spike also from the 2016 graph. Throughout the series, there was a consistency of zero to two times per tweet of mentions, but there were times when there were three times to even four times in a tweet—at the same time, having a tweet reach ten times the number of votes being said in a tweet when it came to very close to election day.



In the second part of the experiment, Text Mining was used using Term Frequency, Inverse Document Frequency, and Term Frequency-Inverse Document Frequency. In the 2016 dataset, there was a total of 324,490 words, while in the 2020 dataset, there was 634,336. When put in perspective, there were 958,826 words across a total of 32,930 tweets.

Keyword	TF	IDF	TF-IDF
Election(2016)	0.0109	1.1451	0.0125
Election(2020)	0.0136	0.9248	0.0126

Vote(2016)	0.0132	0.9554	0.0126
Vote(2020)	0.0201	0.5367	0.0108
Election and Vote(2016)	0.0241	0.3526	0.0085
Election and Vote(2020)	0.0337	0.0189	0.0181

The keyword "election" by itself went up on term frequency and term frequency-inverse document frequency from 2016 to 2020. In contrast, the keyword "vote" had a more significant increase from the keyword election from 2016 to 2020. However, something odd about "vote" was that while it increased on term frequency, it went down when it came to term frequency-inverse document frequency, meaning that it was said more often in a particular tweet than throughout the whole dataset.

Based on the results' outcome, we can see that the 2020 election impacted the general election talk by COVID compared to the non-pandemic election year of 2016. The term frequency of the conversation was higher, while the inverse document frequency went down. This says that the number of times the keywords were mentioned per tweet increased, but the keywords' commonality throughout the dataset went down. However, the term frequency-inverse document frequency did go up by almost 100 points, meaning that a lot of the conversation relevancy did go up.

Aim 2 - Negative Sentiments on Twitter

The second aim of this project is to implement sentiment analysis to detect polarity within a tweet and determine if there was an increase of negative tweets at any point during the pandemic.

Experiments

In order to conclude whether a tweet is negative or not, we must train a model with a dataset of prelabeled tweets. The set of prelabeled tweets used for this portion of the project can be found at https://www.kaggle.com/arkhoshghalb/twitter-sentiment-analysis-hatred-speech. This dataset included over twenty-nine thousand tweets in the training set and over sixteen thousand in the testing set. According to the contributor, a tweet was labeled as hate speech if it contained racist or sexist sentiment.

In order to train the model using these tweets, the tweets must first be cleaned. Using. Python 3.8, each tweet is passed through the "clean_tweet" function. In this function, the mention of the username in a tweet that is a reply is removed, special characters are removed, and the tweet is then tokenized. The tokenization of the tweet allows for individual words of the tweet to be analyzed separately instead of looking at the tweet as a whole. It also allows for stopwords—words that are programmed to be ignored, such as 'the' and 'is'—to be easily filtered out of the tweets. Finally, lemmatization is performed on each word in order to transform all words to their root. After a tweet is cleaned, it is stored in a new column in the training data frame labeled "cleaned_tweet".

```
01. def clean_tweet(tweet):
02.     # remove any @ mentions
03.     remove_mentions = " ".join(filter(lambda x: x[0]!='@', tweet.split()))
04.
05.     # remove special characters
06.     remove_spec_char = re.sub('[^0-9a-zA-Z]', ' ', remove_mentions)
07.     lowercase = remove_spec_char.lower()
08.
09.     # tokenization
10.     tokens = lowercase.split()
11.     stop_words = set(stopwords.words('english'))
12.
13.     # remove stop words (common words programmed to be ignored)
14.     filtered_tweet = [word for word in tokens if not word in stop_words]
15.
16.     # transform words into root words - ex: rocks transforms into rock, demands into demand
17.     filtered_tweet = [lemma.lemmatize(word) for word in filtered_tweet]
18.     filtered_tweet = " ".join(filtered_tweet)
19.     return filtered_tweet
```

Second, the dataset must be split into features and targets. The target for our dataset is the 'negative' label—1 or 0. A tweet is labeled 1 if it is deemed to be negative and it is labeled 0 if it is not. The feature is our cleaned tweet. Two vectorizers are used to transform the cleaned tweet: Countvectorizer and TF-IDF Vectorizer. Countvectorizer, also known as One Hot Encoding, creates vectors of 0's and 1's. Whenever a word is found in the text data, a 1 is placed in the corresponding dimension. If a word is repeated, the count is increased. TF-IDF Vectorizer calculates word frequencies and allows for decisive words to be highlighted.

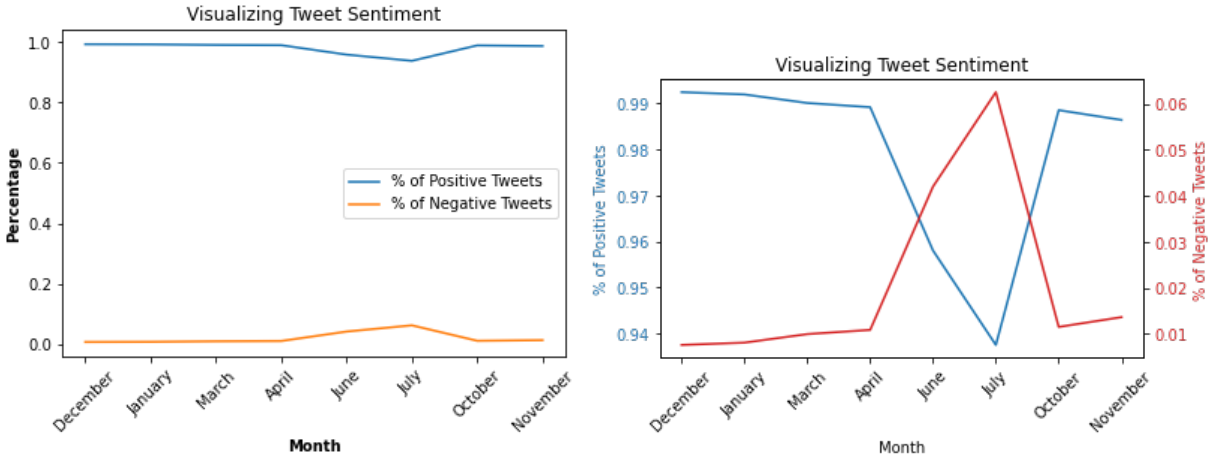
Next, the target and feature can be split into a training set and a testing set. The training set will teach the model how to conclude whether a tweet is negative or not. The testing set will serve as a measure of accuracy. The tweets in the testing set are already labeled, however the trained model will be used on this set and the actual labels will be compared to the predicted labels. Finally, the Logistic Regression model is trained with a random seed of 1.

The model is now ready to be used on the tweets that were collected. In order to view a monthly trend, the analysis was performed on tweets from December 2019 and January, March, April, June, July, October, and November of 2020. In the United States, December and January are pre-COVID months and give an insight on a "normal" number of negative tweets. March and April are the months where lockdowns were beginning. June and July held the highest case counts and perhaps can represent the most trying times in the United States. In October and November sprouted a new normal: lockdowns were no longer necessary, but social distancing and wearing masks were highly encouraged and in some cases required. For each month, the tweet file was imported, the tweets were cleaned in the same manner as the training set, and the model was used to predict whether a tweet was negative or not.

```
# analyze tweets from November 2020
df_nov = pd.read_table('TweetFiles/nov_tweets.csv', sep=',')
df_nov = df_nov.sample(frac=0.25, random_state=1)
nov_tweets = df_nov[['text']].copy()

nov_tweets['cleaned_tweet'] = ""

for i in nov_tweets.index:
    tweet = nov_tweets['text'][i]
    clean = clean_tweet(tweet)
    nov_tweets['cleaned_tweet'][i] = clean

corpus_nov = []
for i in nov_tweets.index:
    corpus_nov.append(nov_tweets.cleaned_tweet[i])

X_test_nov = tfidf.transform(corpus_nov)
y_pred_nov = lr_2.predict(X_test_nov)
nov_tweets['pred_label'] = y_pred_nov

pos_pred_nov = nov_tweets[nov_tweets['pred_label']==0]
num_pos_nov, num_col_nov = pos_pred_nov.shape

neg_pred_nov = nov_tweets[nov_tweets['pred_label']==1]
num_neg_nov, num_col_nov = neg_pred_nov.shape

print("Number of positive tweets in November: " + str(num_pos_nov))
print("Number of negative tweets in November: " + str(num_neg_nov))
```

Results

After predicting the labels using the trained Logistic Regression model, the percentage of positive tweets and negative tweets for each month are calculated and the results are plotted.



On the left hand side is the percentages of both positive and negative tweets on the same scale. On the right hand side, the scale has been adjusted in order to achieve a better sense of the increase and decrease in numbers. The percentage of negative tweets are at the lowest during the pre-COVID months of December and January. There is a slight increase in the percentage of negative tweets in March and April followed by a larger increase during the months of June and July. Lastly, there is a drop in October where the country has reached a level that is closer to normality.

Conclusion

The aim of this portion of the project is to determine if there was an increase of negative tweets at any point during the pandemic. As the results have shown, there was a slight increase in the percentage of negative tweets, especially in the months of June and July. The results for June and July are not surprising as there was a surge in COVID cases in that period of time and quarantine fatigue had likely set in for many people. Some improvements that could be made for future work includes using a different source for pre-labeled tweets, and collecting tweets from months that were not included in the dataset.

Aim 3 - Health Misinformation on Twitter Regarding Covid-19

The aim of this part of the project is to find out if there is a correlation between the number of misinformation about Covid-19 on twitter and the number of Covid-19 positive cases in the US. The motivation for this comes from current concerns about the spread of misinformation coupled with the actions of social media platforms to curb the spread of misinformation.

While the Covid-19 pandemic is still raging, many public health experts including the Director-General of the World Health Organisation (WHO) have bemoaned the growing misinformation about the Covid 19 virus since early in the year, stating that

"We're not just fighting an epidemic; we're fighting an infodemic ".

Their concern about misinformation spreading is due to the effects of misinformation on the general public such as igniting fear, driving societal disaccord, actions leading to direct damage or disruption of hospital services and under-reaction (i.e risky behavior). Most of this misinformation is circulated on social media platforms (such as twitter, facebook and Instagram) on the internet and with many people getting their health information on the internet with the rise in the use of mobile devices, this can have dangerous consequences. For the purpose of this section of the report, misinformation is defined as false assertions regarding Covid-19 and vaccines developed for it.

Social media platforms including Facebook and Twitter have developed a means of checking the spread of misinformation by tagging posts deemed to be inaccurate starting in May 2020. While there is a lot of debate on the morality of this and how the social media giants should not be arbitrators of truth, it has become important that actions needed to be taken to curb the spread of misinformation. Twitter for example is building 'BirdWatch', a crowdsourced system to fight misinformation and instead of 'censoring everything' or 'doing nothing', they provide tags with links to factual information on tweets deemed to be inaccurate. This in itself has not been totally effective as some tags are triggered just by including some key words in the tweet, questioning the ability of machine learning in detecting misinformation or inaccurate information on twitter.

Data

The data set used in this study are live tweets pulled from the Twitter API from October 16 to November 21, 2020 (Roughly 5 weeks). **Tweepy**, a python module for extracting tweets was used for this part of the data collection while the data was saved into a CSV format. The tweets collected were limited to tweets with keyword search of "Corona", "Coronavirus", "Covid-19" and "covid". The code used to generate this is attached in the appendix for Aim 3.

In total, about 300k tweets were collected and out of which 15k tweets were selected at random for each of the 5 weeks. The data collected for each tweet includes the text of the tweet, the user location, tweet ID, date tweet was created, number of retweets, number of the user's followers (the user here refers to the account that tweeted). Figure 1 shows the word cloud from the 75k tweets selected for analysis while figure 2 shows a list of the most occurring hashtags within the tweets.

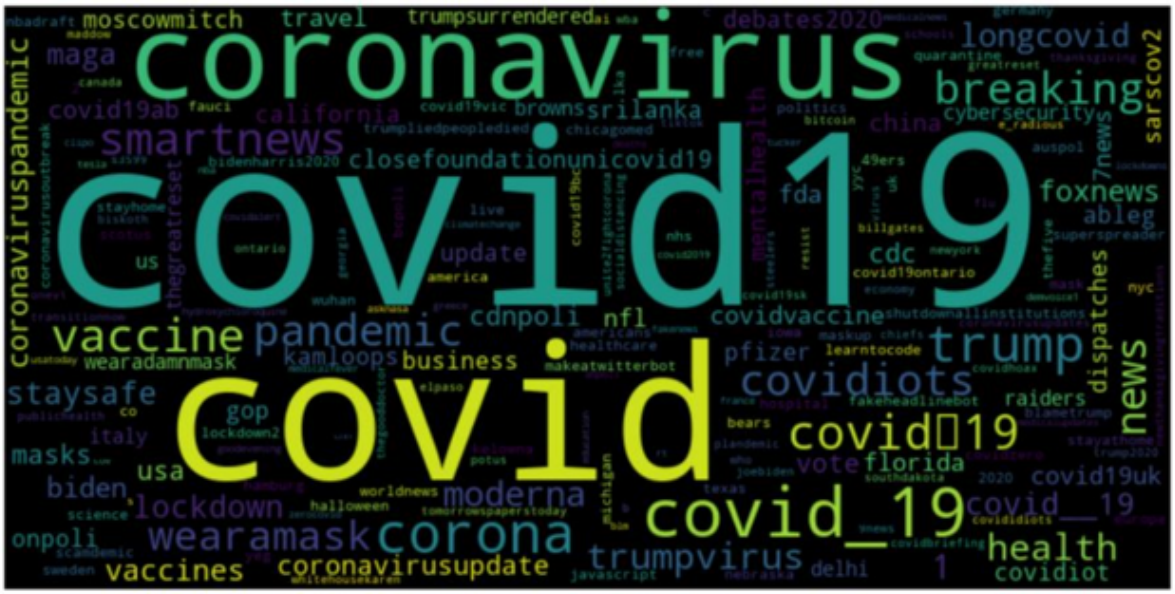


Fig 1. Word cloud of Hashtags from the 75k Dataset

Table 1. Frequency of Top 20 Hashtags within the 75k tweets Dataset

	Hashtags	Frequency
0	covid19	17124
1	covid	10401
2	coronavirus	9791
3	corona	1027
4	trump	970
5	covid_19	920
6	covid—19	831
7	smartnews	760
8	breaking	712
9	pandemic	613
9	pandemic	613
10	news	523
11	covidiots	485
12	wearamask	419
13	vaccine	414
14	blametrump	403
15	trumpvirus	371
16	lockdown	365
17	health	364
18	covid__19	364
19	1	359

Experiments

Python 3 was used in this analysis with Jupyter notebooks running on Anaconda. The analysis was run on a PC with 16GB ram and Intel core i7 processor.

The process followed in the analysis after all the tweets were collected are pre-processing and classification of tweets. Figure 2 shows the process in a flow chart. Pre-processing the tweets involved using the '**preprocessor module**' on python to clean the tweets and tokenize the tweets. The cleaning process removes the emojis and links in the tweets while tokenizing separates the words in the tweet. The tokenized tweets are then lemmatized (i.e words reduced to their root words) using the **NLTK** (natural language toolkit) module called **WordNetLemmatizer** on python. The python codes for these processes are included in the appendix.

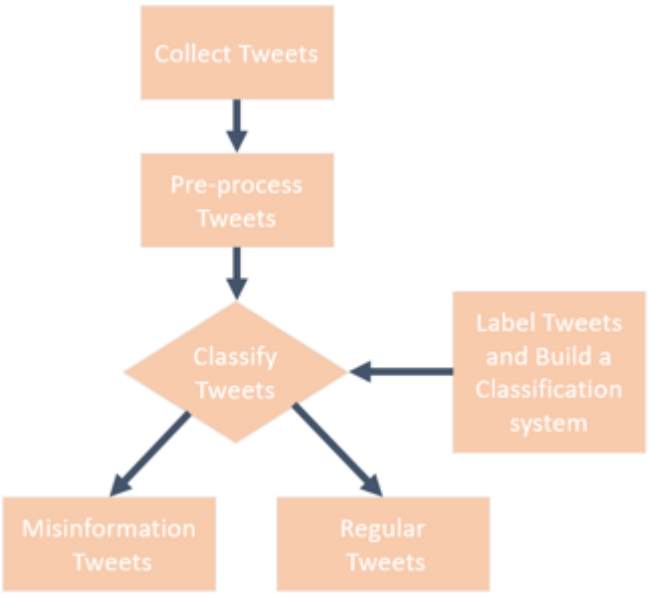


Fig 2. Analysis Steps

To classify the tweets, two (2) supervised machine learning (ML) algorithms are trained to identify misinformation tweets using labelled data. 54 hand labelled misinformation tweets and 54 hand labelled regular tweets were used to build this classification model. The labelling was based on information provided on factcheck.org, politifact.com, fullfact.org and snopes.com. For example, below is a tweet in the labelled data;

"The same is true for Pfizer as well. Moderna's and Pfizer's vaccines are leveraging mRNA technology. This will change our DNA! Please listen to Fr. Altier's homily about this!"

The tweet above falsely claims that the Covid 19 Vaccine alters the DNA. This has been debunked by many fact checking organizations. A tweet like this is labelled as misinformation. Figure 3 shows an article from fullfact.org debunking the false assertion.

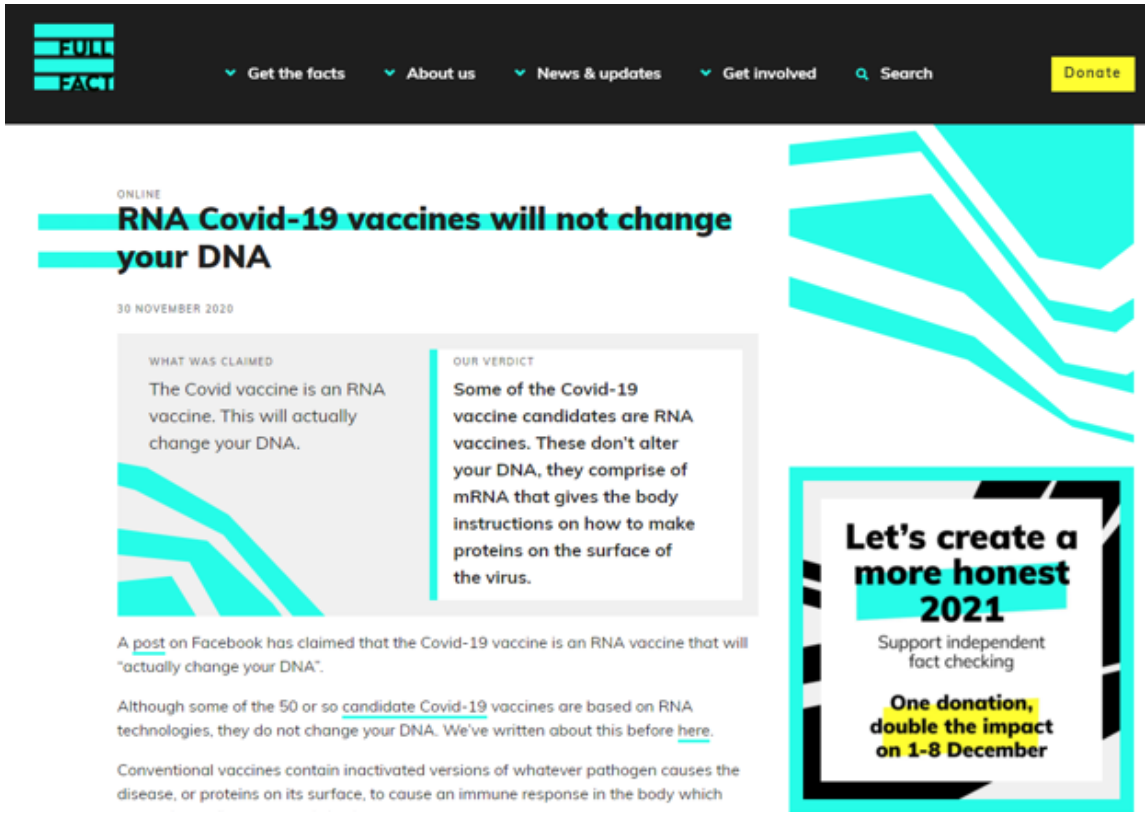


Fig.3 Debunked claims from fullfact.org (Accessed: December 4, 2020)

Figure 4 and 5 shows the confusion matrix of the Gaussian Naive Bayes model and Random Forest Classifier ML tool developed. The Sklearn tools were used for this part of the project.

```
[[ 4  5]
 [ 2 11]]
                    precision    recall  f1-score   support

     Misinformation       0.67      0.44      0.53         9
Non-Misinformation       0.69      0.85      0.76        13

        micro avg       0.68      0.68      0.68        22
        macro avg       0.68      0.65      0.65        22
     weighted avg       0.68      0.68      0.67        22

0.6818181818181818
```

Fig 4. Confusion Matrix, Classification Report and Accuracy of Gaussian Naive Bayes model

```
[[ 7  2]
 [ 2 11]]
                    precision    recall  f1-score   support

    Misinformation       0.78      0.78      0.78         9
Non-Misinformation       0.85      0.85      0.85        13

         micro avg       0.82      0.82      0.82        22
         macro avg       0.81      0.81      0.81        22
      weighted avg       0.82      0.82      0.82        22

0.8181818181818182
```

Fig 5. Confusion Matrix, Classification Report and Accuracy of Random Forest Classifier

The random forest classifier method was chosen to classify the entire 75000 tweets based on comparing the performance of the two classifiers.

Another ML classification tool was trained to classify tweets as either 'Positive' or 'Negative using a corpus of 14000 pre-labelled positive and negative tweets from NLTK package (available online with 20k tweets and downloaded using the line of code '*from nltk.corpus import twitter_samples'*), using the Naïve Bayes classifier (shown in Appendix: Python Code for Positive and Negative Tweets). The accuracy of the model was 0.99533333. The Number of Negative tweets in the 75k tweet dataset were 36200 and Positive tweets were 38,800.

Results

The classification of the tweets into either positive or negative tweets indicate that throughout the five-week period there were almost as many positive tweets as there were negative tweets. Figure 6 shows the most frequent words in the positive tweets and those on the negative tweets. Asides from the key word search words, 'case', 'new', 'people', 'positive' and 'trump' were the highest words in the positive tweets while 'people', 'get', 'vaccine', 'trump' and 'like' are amongst the highest negative words. One inference that can be drawn from this is that possibly there are more negative tweets about the Covid 19 vaccine than there are positive tweets based on the position of 'vaccine' in the rank. Together with Table 1 showing the most frequent hashtags, vaccines are one of the 'hot topics' for conversations, especially as most vaccine candidates are currently being checked for approval in the US.

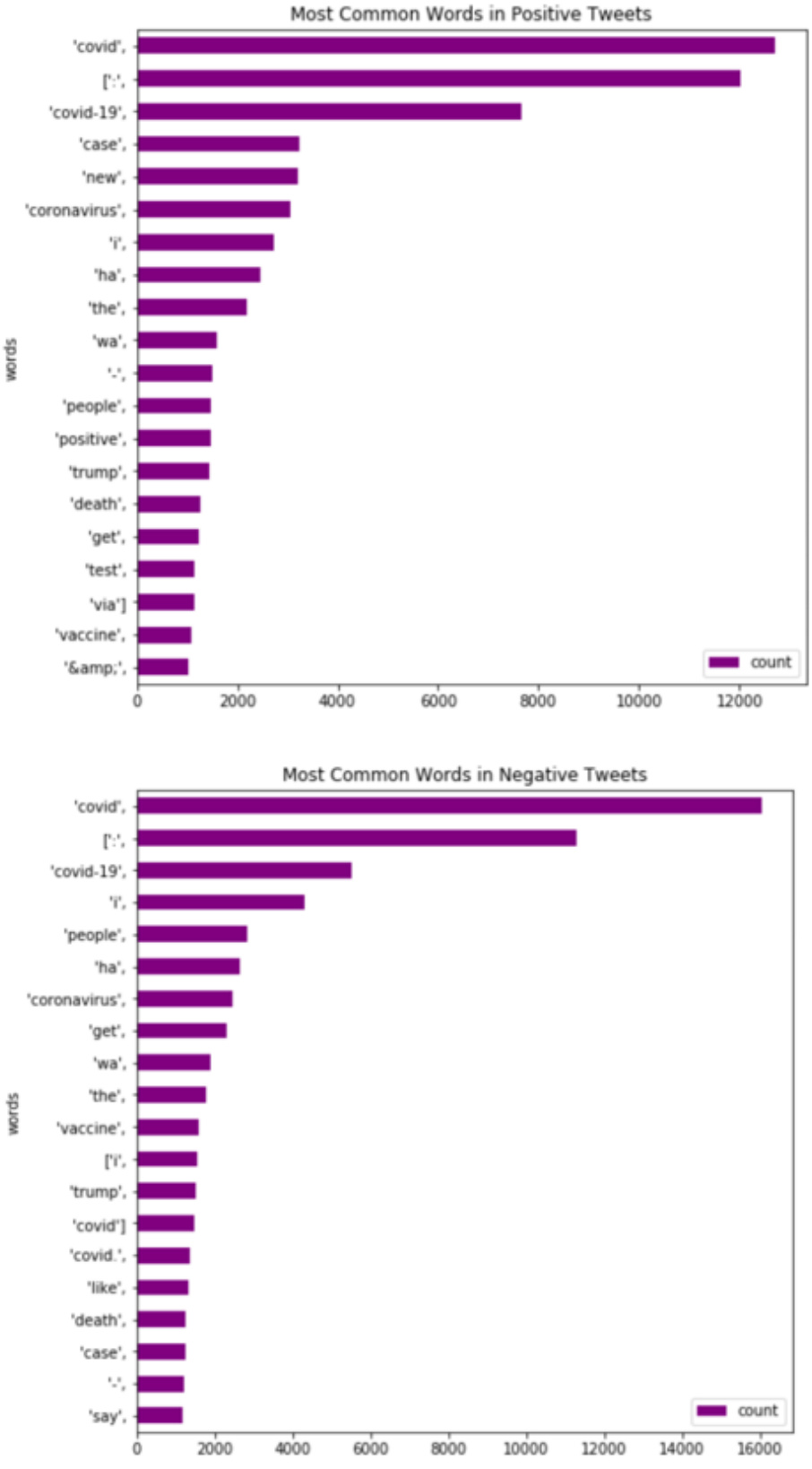


Fig 6. Most Common words in Positive and Negative tweets

One of the options considered in this study was to find out if there was a correlation between negative tweets and misinformation, so the hand labelled tweets were classified as either positive or negative using the ML model discussed in the experiment section. Some part of the table are shown in Table 2. The data indicates that not all misinformation tweets are negative and not all Non-misinformation tweets or regular tweets were positive. This further shows that a simple positive or negative sentiment analysis would not be enough in predicting misinformation.

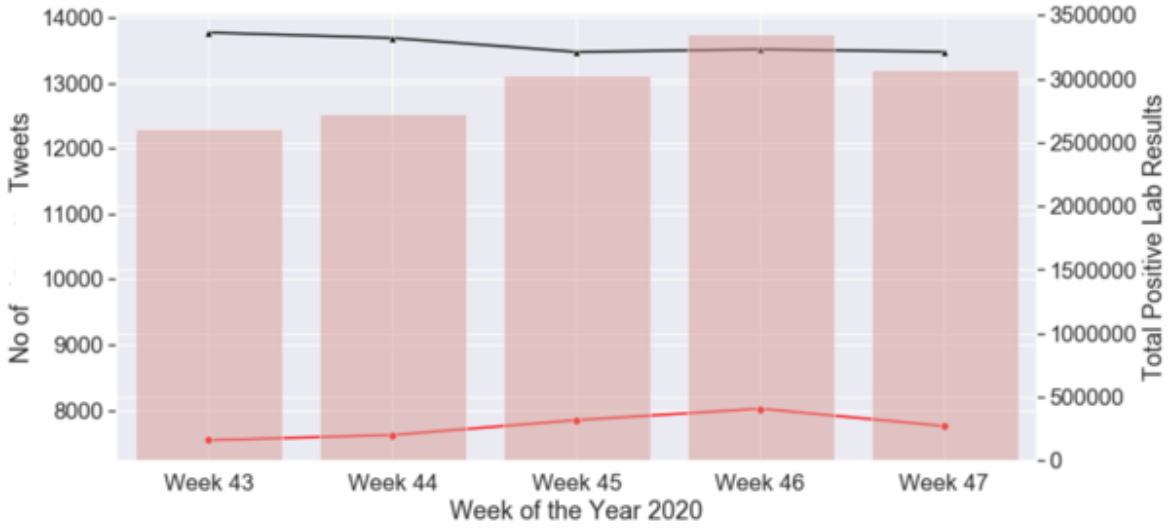
Table 2. Classification of Labelled Tweets

	tweet	classification	classify
0	the covid vaccine is an rna vaccine this will ...	Misinformation	Negative
1	you are happy to be able to kill your people w...	Misinformation	Negative
2	moderna vaccine is not morally produced unborn...	Misinformation	Negative
3	the same is true for pfizer as well modernas a...	Misinformation	Negative
4	check out nenas video tiktokagenda 21 2030 fri...	Misinformation	Positive
5	control the vaccines literally change our dna ...	Misinformation	Negative
6	the corona vaccine is dna altering i will neve...	Misinformation	Negative
7	an experimental rna vaccine that alters your ...	Misinformation	Positive
97	i get called pretty more w this mask on then i...	Non-Misinformation	Positive
98	remember when some people kept saying the coro...	Non-Misinformation	Negative
99	taking medical advice from experts to not spre...	Non-Misinformation	Positive
100	people are so inconvenienced by wearing masks ...	Non-Misinformation	Negative
101	im not uneducated the president has covid im n...	Non-Misinformation	Negative
102	i really dont want to hear yalls conspiracy th...	Non-Misinformation	Negative
103	news needs to point out that when republican c...	Non-Misinformation	Negative
104	effects of our restrictions due to the corona...	Non-Misinformation	Negative
105	i have a proposal lets all go vegan at least f...	Non-Misinformation	Negative
106	can i be frank most other countries have a cen...	Non-Misinformation	Positive

The classification model developed to classify tweets as either misinformation or non-misinformation (regular tweets) classified about 95% of the entire dataset as misinformation. This value is high and gives a similar scenario to when twitter had information/warning labels(tags) on any tweet with keywords ‘Holocaust’ or ‘Election Fraud’. Table 3 is a summary of the results showing the number of negative tweets and misinformation tweets over each of the five (5) weeks data while a column includes the number of covid 19 positive cases from lab tests across the US. The data was sourced from the CDC official website. Figure 7 is a plot of the data in Table 3.

Table 3. Total number of misinformation tweets, negative tweets and positive covid 19 lab results in the US.

	Week of the Year 2020	No of Misinformation Tweets	No of Negative Tweets	Total Positive Lab Results
0	Week 43	13769	7539	2611246
1	Week 44	13685	7619	2728875
2	Week 45	13473	7846	3032328
3	Week 46	13512	8021	3356170
4	Week 47	13475	7755	3069436



*Black line plot – No of misinformation tweets

**Red line plot – No of Negative tweets

*** Bar chart plot – Number of positive Covid 19 lab results across the US

Figure 7. Plot of Covid positive cases, Negative Tweets and Misinformation tweets

Figure 7 indicates that there is almost no visible correlation between the number of misinformation on twitter and the number of positive Covid 19 lab results. However, the trend of the negative tweets match with the number of positive Covid 19 test results as both increase from week 43 till week 46 before dropping at week 47.

Conclusion

This part of the project has looked into the possibility of a correlation between the number of misinformation on twitter and the positive Covid 19 lab results within the US. The data analysis suggests no correlation but a simple correction was obtained for negative tweets and the positive Covid 19 lab results.

However, reasons for the lack of correlation between the number of misinformation tweets and the number of positive covid-19 test results as shown in this study could vary from;

1. The model for classification was not the best due to the limited number of training data. This could be improved by having more training dataset, using a different ML classifier or improving the current ML classifier.

2. Looking at the data for the US as a whole or weekly could constitute a case of Simpson's paradox, while breaking it down to state levels or daily trends might show better correlation.

3. The tweets collected might not be representative of all the states in the US. This could be looked into to ensure adequate representation.

These three (3) points can form a basis for further study on the subject together with a look into an analysis of the emojis in tweets, as this could be another key to unlocking the detection of misinformation using ML models.

Finally, detecting health misinformation using ML models can be a difficult task given that there is no generally acceptable definition for misinformation and no totally faultless source of factual information, coupled with the changing nature of facts especially in the field of health and novel viruses. This notwithstanding, research, studies and methods to make this a reality would continue as the effects of health misinformation through social media continues to be a threat to public health and safety across the world.

Aim 4 - Impact of Covid-19 on Human Mobility

The last aim of this project is to look into the geotagged tweet to understand the impact of COVID on human mobility. Understanding human mobility is very important in the case of many applications from disease prediction to communication networks. It has fundamental applications from urban planning (Noulas et al. 2012) to human and electronic virus prediction (Balcan et al. 2009; Wang et al. 2009; Tizzoni et al. 2014) and traffic and population forecasting (Wilson and Bell 2004). Twitter relies on publicly available data and provides high resolution positioning when users opt to geotag their tweets with their current location. Most of the studies to understand human mobility patterns have been done on private and low resolution data, such as mobile phone data. As covid has impacted us through many ways, it certainly changed the pattern of general human mobility. In this study, it was attempted to understand this change in the U.S. with geotagged tweets to have an idea about the potentiality of twitter as a proxy.

Data Collection and Description:

A Twitter Developer Account was created using Twitter Apps (apps.twitter.com/) to retrieve data through Twitter Streaming API (Application Programming Interfaces). Python programming language was used to collect the data and associated Python libraries have been used. As the main focus of this study is geo-tagged tweets generated in the USA, the location-based data collection method has been used which has the ability to produce a more suitable and reliable dataset that serves the goal of the study. As a result, tweets from North America and its surrounding area, confined by approximately (14.4439373, -18.324324) and (71.218493, -169.3692058) coordinates, are collected using a location-bounding box which covered USA, Canada, Mexico, Cuba, Puerto Rico, and part of Guatemala and Greenland (Fig. 8) to assure that all the tweet generated in usa can be captured for two different period of time; 1) pre-covid data (Dec-Jan dataset: Dec 16, 2019- Jan 4, 2020) for around 20 days and 2) covid data (Mar-Apr dataset: Mar 13, 2020- Apr 11, 2020) for around 26 days. The raw data were saved in a .json file. The Dec-Jan dataset was of around 46.6 GB containing approximately 12.9M tweets and the Mar-Apr dataset has a size of around 84.5 GB with 23.8 M tweets.

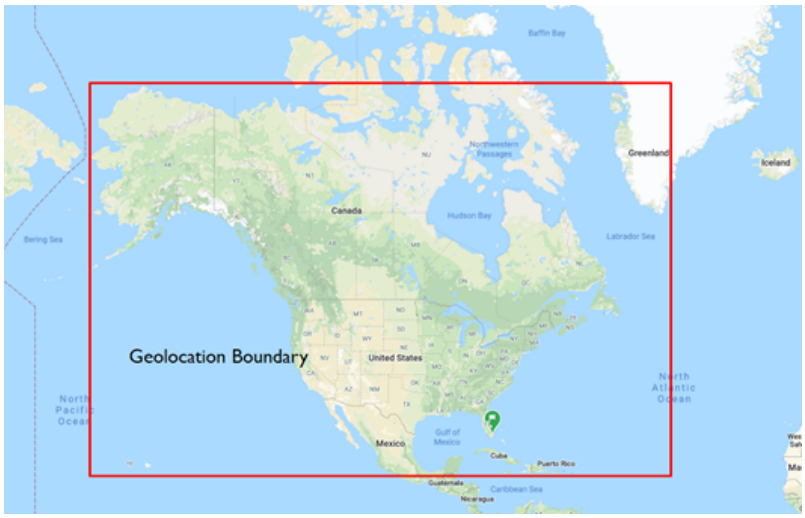


Figure 8. Bounding box for data collection from North America

Methodology:

Data cleaning and preprocessing:

For the purpose of this we focused on the geotagged tweets as the geolocation information of the user is an important information in the study. The key steps of the data cleaning and preprocessing is presented in Fig. 9. There are two major steps in the cleaning and preprocessing of the data. In the first step all data

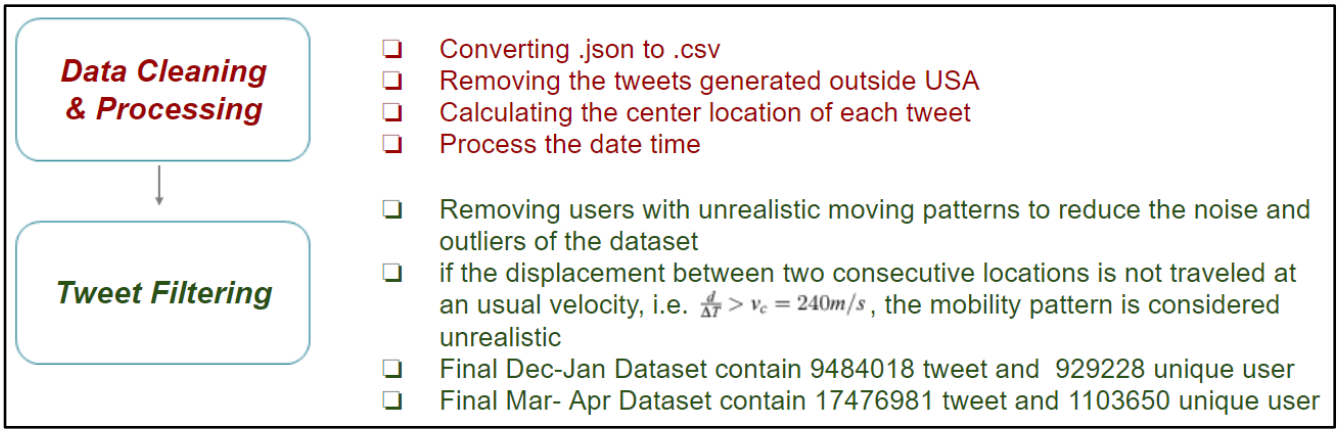


Figure 9. Key steps in data cleaning and preprocessing.

has been converted to a .csv file from .json. As our focused region in the USA, the tweet generated from outside the USA was removed. In this dataset, each tweet record has a geotag and a timestamp indicating where and when the tweet was posted. Based on this information we are able to construct a user's location history denoted by a sequence $L = \{(x_i, y_i, t_i)\}$. In the next step, the dataset has been filtered. There are a lot of unusual users which create noise and contribute contamination to the results. In this study, if the displacement between two consecutive locations is not traveled at

an usual velocity, i.e. greater than 240 m/s, we consider the mobility pattern is unrealistic and remove those users from the dataset. The following study is then based on the filtered dataset which contains 9484018 tweets and 929228 unique users for Dec-Jan dataset and 17476981 tweets and 1103650 unique users for Mar-Apr dataset.

Result and Discussion

Understanding the individual mobility pattern is a very crucial task. There are several characteristics of human mobility. In this study only displacement distribution and radius of gyration have been studied. The movement patterns of individuals were first characterised by analysing their sequences of geotagged tweets. Based on these sequences, the moving distances for individual trips were studied over the long-term.

Displacement distribution (d)

The first important characteristic in human mobility patterns is the displacement distribution, where d is the distance between a user's two consecutive reported locations. Fig. 10 shows the comparative analysis of the daily mean distance travelled by individual users between the two dataset for the last consecutive 18 days. The heterogeneity in the mean travel distances for the pre-covid dataset represents the usual

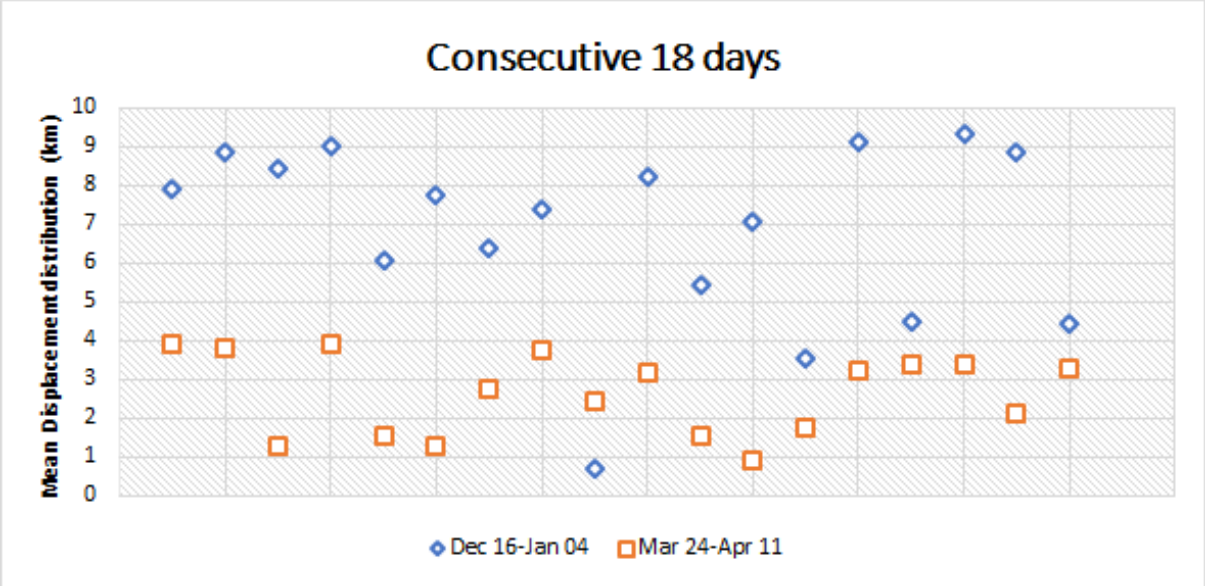


Figure 10. Daily mean displacement distribution between pre-covid and covid dataset.

nature of human mobility. No uniform distribution pattern was observed from the pre-covid dataset. On the other hand, a relatively stable uniform pattern was observed from the covid dataset. The displacement distribution was going more towards uniformity over time. By looking at values of the last five days from the covid dataset, it can be suggested that the individual mobility was

becoming uniform gradually. Fig. 10 tells that the users normally made heterogenous trips of higher distance before covid which was different from the travel behavior of users during covid. During covid users started to make quite uniform trips of shorter distance which implies that the covid has impacted human travel in many ways.

Radius of gyration (r_g)

To explore the heterogeneity of mobility among individuals, we study the radius of gyration rg, which is another important characteristic of human mobility that quantifies the spatial stretch of an individual trajectory or the traveling scale of an individual. The radius of gyration for an individual can be calculated by $r_g = \sqrt{\frac{1}{n} \times \sum_i \quad (r_i - r_c)^2}$, where r_i is the individual's i-th location and $r_c = \frac{1}{n}\sum_i \quad r_i$ is the geometric center of the trajectory and n is the number of locations in the trajectory. Fig. 11 shows the comparison between pro-covid and covid dataset for the daily mean radius of gyration for the last consecutive 18 days. The distribution of the radius of gyration rg over the whole population has a similar shape as observed in the displacement distribution. Fig. 11 tells us that before covid people used to travel more far from home than the period of covid. The heterogeneity has been seen in the pre-covid dataset, but during covid people reduced their traveling extent and gradually followed the uniform stable pattern. This indicate the impact of covid as they adopt various precautionary measures, i.e. work-from-home, social distancing, and doing online shopping activities.

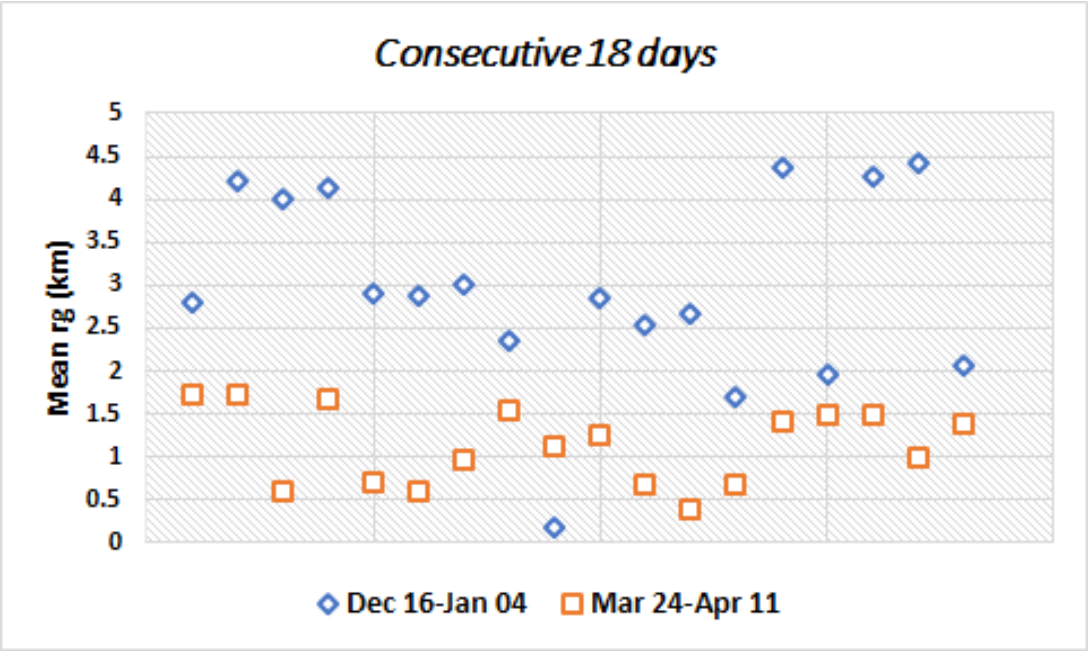


Figure 11. Daily mean radius of gyration between pre-covid and covid dataset.

Conclusion

The aim of this portion of the project is to use twitter as a proxy to understand the impact of COVID on human mobility. At the beginning of this pandemic in this country, people were almost aware of the precautionary measures to save themselves from COVID. The government also declared a national emergency to stop the spread describing the precautionary steps, i.e. social distancing, online study program, online shopping, work-from-home etc. All these measures impacted the general human mobility greatly. Measuring these changes is a crucial task for the researcher. In this part of the study the potentiality of twitter as a proxy was accessed to understand the changes in human mobility for COVID. The results show quite interesting positive outputs regarding the potentiality of twitter understanding human mobility pattern. From fig. 10 and fig. 11, it can be easily observed that 1) people make trips of short distance more now-a-days than before; 2) people traveling boundary from home is shorter than before; 3) people's traveling behaviors are becoming more uniform gradually over time reflects response of the peoples' adoption of the precautionary measures from COVID.

General Conclusion

This study has considered the changes in Twitter trends in the US due to Covid-19 intending to show the impact of Covid-19 on the 2020 elections in comparison to the 2016 elections, changes in tweet sentiments along the year, checking the correlation between health misinformation on Twitter and positive covid cases and the impact of Covid-19 on human mobility.

Based on the dataset and analysis carried out, results show that the general election talk impacted the 2020 election on COVID compared to the non-pandemic election year of 2016. A rise in negative tweets that coincided with periods of surges in COVID cases and quarantine lockdowns was detected, revealing that societal trends can be gleaned from the Twitter feed. The study could not find any correlation between the number of misinformation tweets and Covid-19 positive cases but instead found a correlation between negative tweets and the Covid-19 positive cases. The study also showed how Twitter can be used to judge human mobility and found a reduction in mobility during the Covid-19 period/lockdowns.

Finally, the project allowed the team to use the skills learned in class on real-world data.

Website: https://twitteranalysis.wixsite.com/twitteranalysis

Appendix

Appendix for Aim 1

Python Code for the Twitter API Data collection.

```
from searchtweets import ResultStream, gen_rule_payload, load_credentials
from searchtweets import collect_results
import csv

premium_search_args = load_credentials("twitter_keys.yaml", yaml_key="search_tweets_fullarchive_development2", env_overwrite=False)

with open('electionNov07-2016-2.csv', 'w', newline='') as csvfile:
   writer = csv.writer(csvfile)
   writer.writerow(["Language", "date_created", "Tweet", "num_likes", "num_retweets", "Followers", "Quote Count",
                    "User Name", "ID", "Poll Options"])

   rule = gen_rule_payload("vote lang:en place_country:US", results_per_call=100,
                       from_date="2016-11-07 03:01", to_date="2016-11-07 23:59")
   tweets = collect_results(rule,
                        max_results=100,
                        result_stream_args=premium_search_args)

   for t in tweets:
       print(t)
       writer.writerow([t.lang, t["created_at"], t.all_text, t.favorite_count, t.retweet_count, t.follower_count,
                        t.quote_count, t.screen_name, t.id, t.poll_options])
```

Python Jupyter Code

```
print("2020 Data")
print("Election was tweeted:",dataset.Tweet.str.count("election").sum())
print("Vote was tweeted:",dataset.Tweet.str.count("vote").sum())
print("2016 Data")
print("Election was tweeted:",data2016.Tweet.str.count("election").sum())
print("Vote was tweeted:",data2016.Tweet.str.count("vote").sum())
cols_plot = ['Num_Vote', 'Num_Election']
axes = dataset[cols_plot].plot(marker='.', alpha=0.5, linestyle='None', figsize=(11, 9), subplots=True)
for ax in axes:
    ax.set_ylabel('Times Mentioned per Tweet')

plt.savefig('TIMESERIES2020.png', dpi=500)
cols_plot = ['Num_Vote', 'Num_Election']
axes = data2016[cols_plot].plot(marker='.', alpha=0.5, linestyle='None', figsize=(11, 9), subplots=True)
for ax in axes:
    ax.set_ylabel('Times Mentioned per Tweet')
plt.savefig('TIMESERIES2019.png', dpi=500)
ni = dataset.Tweet.str.split().str.len()
wordsCountInFile=ni.sum()
num2016 = data2016.Tweet.str.split().str.len()
wordsCountInFile2016=num2016.sum()
print("There is a total of",wordsCountInFile2016, "words in the 2016 file")
print("There is a total of",wordsCountInFile, "words in the 2020 file")
diffWordCount=wordsCountInFile-wordsCountInFile2016
print("Both files have the same amount of tweets 16465, but there was an increase of",diffWordCount, "words from 2016 to 2020")
wordsInBoth=wordsCountInFile2016+wordsCountInFile
numOfDoc=16465*2
print(wordsInBoth)
print(numOfDoc)
import math
electionCount=dataset.Tweet.str.count("election").sum()
tfElection=round((electionCount/wordsInBoth),4)
print("The TF of election-2020 is",tfElection)
idfElection= round((math.log(numOfDoc/electionCount)),4)
tfIDFelection= round((idfElection*tfElection),4)
print("The IDF of election-2020 is",idfElection)
print("The TF-IDF weight-2020 is",tfIDFelection)
voteCount=dataset.Tweet.str.count("vote").sum()
tfVote=round((voteCount/wordsInBoth),4)
print("The TF of vote-2020 is",tfVote)
idfVote= round((math.log(numOfDoc/voteCount)),4)
tfIDFvote= round((tfVote*idfVote),4)
```

```
print("The IDF of vote-2020 is",idfVote)
print("The TF-IDF weight-2020 is",tfIDFvote)
everyCount=electionCount+voteCount
everyPercent=round((everyCount/wordsInBoth),4)
print("The TF of election talk in 2020 is",everyPercent)
idfEvery= round((math.log(numOfDoc/everyCount)),4)
tfIDFevery= round((everyPercent*idfVote),4)
print("The IDF of election talk in 2020 is",idfEvery)
print("The TF-IDF weight-2020 is",tfIDFevery)
electionCount2016=data2016.Tweet.str.count("election").sum()
electionPercent2016=round((electionCount2016/wordsInBoth),4)
print("The TF of election-2016 is",electionPercent2016)
idfElection2016= round(math.log(numOfDoc/electionCount2016),4)
tfIDFelection2016= round((electionPercent2016*idfElection2016),4)
print("The IDF of election-2016 is",idfElection2016)
print("The TF-IDF weight-2016 is",tfIDFelection2016)
voteCount2016=data2016.Tweet.str.count("vote").sum()
votePercent2016=round((voteCount2016/wordsInBoth),4)
print("The TF of vote-2016 is",votePercent2016)
idfVote2016= round((math.log(numOfDoc/voteCount2016)),4)
tfIDFvote2016= round((votePercent2016*idfVote2016),4)
print("The IDF of vote-2016 is",idfVote2016)
print("The TF-IDF weight-2016 is",tfIDFvote2016)
count2016=voteCount2016+electionCount2016
percent2016=round((count2016/wordsInBoth),4)
print("The TF of election talk in 2016 is",percent2016)
idfEvery2016= round((math.log(numOfDoc/count2016)),4)
tfIDFevery2016= round((percent2016*idfEvery2016),4)
print("The IDF of election talk in 2016 is",idfEvery2016)
print("The TF-IDF weight-2016 is",tfIDFevery2016)
```

Appendix for Aim 2

Python Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import re
import seaborn as sns
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
lemma = WordNetLemmatizer()
from nltk import FreqDist
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, f1_score, accuracy_score

# import training datasets
df_train = pd.read_table("train.csv", sep=',')
df_train = df_train[['label', 'tweet']].copy()

def clean_tweet(tweet):
    # remove any @ mentions
    remove_mentions = " ".join(filter(lambda x: x[0]!='@', tweet.split()))

    # remove special characters
    remove_spec_char = re.sub('[^0-9a-zA-Z]', ' ', remove_mentions)
    lowercase = remove_spec_char.lower()

    # tokenization
    tokens = lowercase.split()
    stop_words = set(stopwords.words('english'))

    # remove stop words (common words programmed to be ignored)
    filtered_tweet = [word for word in tokens if not word in stop_words]

    # transform words into root words - ex: rocks transforms into rock, demands into demand
    filtered_tweet = [lemma.lemmatize(word) for word in filtered_tweet]
    filtered_tweet = " ".join(filtered_tweet)
    return filtered_tweet

# add a new column to the dataframe to hold the cleaned tweets
cleaned_tweets = []
for i in df_train.index:
    tweet = df_train['tweet'][i]
    clean = clean_tweet(tweet)
    cleaned_tweets.append(clean)
df_train['cleaned_tweet'] = cleaned_tweets
```

```
corpus = []
for i in df_train.index:
    corpus.append(df_train['cleaned_tweet'][i])

# CountVectorizer is also known as One Hot Encoding
# CountVectorizer creates vectors that have a dimensionality equal to the number of words in our vocabulary
# whenever a word is found in the text data, a 1 is placed in the corresponding dimension
# if a word is repeated, we increase the count
# if a word is not found, the word count remains at 0
# stopwords are ignored

cv = CountVectorizer(stop_words=stopwords.words('english'))
cv.fit(corpus)

# split features from targets in datasets:
X = cv.transform(corpus).toarray()
y = df_train.iloc[:, 0].values

X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

lr_1 = LogisticRegression(random_state=1)
lr_1.fit(X_train, y_train)
y_pred = lr_1.predict(X_test)

# tf-idf is term frequency (tf) * inverse data frequency (idf)
# term frequency is the number of times a term appears in a text divided by the total number of words in the text
# inverse data frequency is log(number of documents / number of documents containing the word w)
# idf determines the weight of the word

# min_df = 20 in order it ignore terms that appear less than 20 times
tfidf = TfidfVectorizer(ngram_range=(1, 2), min_df=20, stop_words=stopwords.words('english'))
X2 = tfidf.fit_transform(corpus)
X2_train, X2_test, y2_train, y2_test = \
    train_test_split(X2, y, test_size=0.3, random_state=1)

lr_2 = LogisticRegression(random_state=1)
lr_2.fit(X2_train, y2_train)
y2_pred = lr_2.predict(X2_test)
```

```
# analyze tweets from April 2020
df_apr = pd.read_table('TweetFiles/apr_tweets.csv', sep=',')
df_apr = df_apr.sample(frac=0.25, random_state=1)
apr_tweets = df_apr[['text']].copy()

apr_tweets['cleaned_tweet'] = ""

for i in apr_tweets.index:
    tweet = apr_tweets['text'][i]
    clean = clean_tweet(tweet)
    apr_tweets['cleaned_tweet'][i] = clean

corpus_apr = []
for i in apr_tweets.index:
    corpus_apr.append(apr_tweets.cleaned_tweet[i])

X_test_apr = tfidf.transform(corpus_apr)
y_pred_apr = lr_2.predict(X_test_apr)
apr_tweets['pred_label'] = y_pred_apr

pos_pred_apr = apr_tweets[apr_tweets['pred_label']==0]
num_pos_apr, num_col_apr = pos_pred_apr.shape

neg_pred_apr = apr_tweets[apr_tweets['pred_label']==1]
num_neg_apr, num_col_apr = neg_pred_apr.shape

print("Number of positive tweets in April: " + str(num_pos_apr))
print("Number of negative tweets in April: " + str(num_neg_apr))
```

Appendix for Aim 3

Python Code for Tweet Collection on Twitter API

```
from __future__ import absolute_import, print_function

import tweepy

from tweepy.streaming import StreamListener

from tweepy import OAuthHandler

from tweepy import Stream

import dataset

from sqlalchemy.exc import ProgrammingError


consumer_key = open("API_KEY.txt").read().replace("\n", "")

consumer_secret = open("API_SECRET_KEY.txt").read().replace("\n", "")

access_token = open("ACCESS_TOKEN.txt").read().replace("\n", "")

access_token_secret = open("ACCESS_TOKEN_SECRET.txt").read().replace("\n", "")



class StdOutListener(StreamListener):

    def on_status(self, status):


            print(status.text)


            if status.retweeted:

            return


            api = tweepy.API(auth)
```

```
id_str = status.id_str

#status1 = api.id_str

created = status.created_at

text = status.text

fav = status.favorite_count

name = status.user.screen_name

description = status.user.description

loc = status.user.location

user_created = status.user.created_at

followers = status.user.followers_count

retweets = status.retweet_count

#retweets = status1.retweet_count

table = db['myTweets']

try:

table.insert(dict(

id_str=id_str,

created=created,

text=text,

fav_count=fav,

user_name=name,

user_description=description,

 user_location=loc,

user_created=user_created,

user_followers=followers,

retweet_count=retweets,
```

```
            ))
            except ProgrammingError as err:
            print(err)
    def on_error(self, status_code):
            if status_code == 420:
            return False



if __name__ == '__main__':
    db = dataset.connect("sqlite:///tweets_countrywide15.db")
    # tweets_countrywide2.db exceeded the 1 million limit of csv lines so we include a number
            l: StdOutListener = StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    stream = Stream(auth, l)
    stream.filter(languages=["en"], locations=[-125.11, 24.36, -66.66, 49.26], track=['Covid-19', 'corona',
                                                        'coronavirus', 'covid',
                                                        '#coronavirus', '#Covid-19',
                                                        '#Corona', '#covid'])
```

Python Code for Positive and Negative Tweets

```
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import twitter_samples, stopwords
from nltk.tag import pos_tag
from nltk.tokenize import word_tokenize
from nltk import FreqDist, classify, NaiveBayesClassifier

import re, string, random
```

```
def remove_noise(tweet_tokens, stop_words = ()):

    cleaned_tokens = []

    for token, tag in pos_tag(tweet_tokens):
    token = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|'\
                '(?:%[0-9a-fA-F][0-9a-fA-F]))+','', token)
    token = re.sub("(@[A-Za-z0-9_]+)","", token)

    if tag.startswith("NN"):
        pos = 'n'
    elif tag.startswith('VB'):
            pos = 'v'
    else:
            pos = 'a'

    lemmatizer = WordNetLemmatizer()
    token = lemmatizer.lemmatize(token, pos)

    if len(token) > 0 and token not in string.punctuation and token.lower() not in stop_words:
        cleaned_tokens.append(token.lower())
    return cleaned_tokens

def get_all_words(cleaned_tokens_list):
    for tokens in cleaned_tokens_list:
    for token in tokens:
            yield token

def get_tweets_for_model(cleaned_tokens_list):
    for tweet_tokens in cleaned_tokens_list:
    yield dict([token, True] for token in tweet_tokens)

if __name__ == "__main__":

    positive_tweets = twitter_samples.strings('positive_tweets.json')
    negative_tweets = twitter_samples.strings('negative_tweets.json')
    text = twitter_samples.strings('tweets.20150430-223406.json')
    tweet_tokens = twitter_samples.tokenized('positive_tweets.json')[0]

    stop_words = stopwords.words('english')

    positive_tweet_tokens = twitter_samples.tokenized('positive_tweets.json')
   negative_tweet_tokens = twitter_samples.tokenized('negative_tweets.json')

   positive_cleaned_tokens_list = []
   negative_cleaned_tokens_list = []

    for tokens in positive_tweet_tokens:
      positive_cleaned_tokens_list.append(remove_noise(tokens, stop_words))
```

```
for tokens in negative_tweet_tokens:
  negative_cleaned_tokens_list.append(remove_noise(tokens, stop_words))

all_pos_words = get_all_words(positive_cleaned_tokens_list)

freq_dist_pos = FreqDist(all_pos_words)
print(freq_dist_pos.most_common(10))

positive_tokens_for_model = get_tweets_for_model(positive_cleaned_tokens_list)
negative_tokens_for_model = get_tweets_for_model(negative_cleaned_tokens_list)

positive_dataset = [(tweet_dict, "Positive")
                    for tweet_dict in positive_tokens_for_model]

negative_dataset = [(tweet_dict, "Negative")
                    for tweet_dict in negative_tokens_for_model]

dataset = positive_dataset + negative_dataset

random.shuffle(dataset)

train_data = dataset[:7000]
test_data = dataset[7000:]

classifier = NaiveBayesClassifier.train(train_data)

print("Accuracy is:", classify.accuracy(classifier, test_data))
```

Python Code to Classify Tweets as Negative or Positive

```
P = []
for i in range(len(Alltweets1)):
  D = Alltweets1.loc[i,('tokenized')]
  F = remove_noise(word_tokenize(D))
  m = classifier.classify(dict([token, True] for token in F))
  P.append(m)
df = pd.DataFrame(P, column = 'classify')
df
#Alltweets now contains everything
Alltweets.join(df)
P.count('Negative')
```

List of Negative words and list of Positive words in tweets

```
# Create an empty list
```

```
    Row_list_positive =[]
    # Iterate over each row
    for index, rows in the_positive_tweets.iterrows():
        # Create list for the current row
        my_list =rows.no_stop_words
        # append the list to the final list
        Row_list_positive.append(my_list)
    Row_list_negative =[]
    # Iterate over each row
    for index, rows in the_negative_tweets.iterrows():
        # Create list for the current row
        my_list =rows.no_stop_words
        # append the list to the final list
        Row_list_negative.append(my_list)
```

Plot Negative and Positive Common Words

```
## Create three Counter objects to store positive, negative and total counts
positive_counts = Counter()
negative_counts = Counter()
total_counts = Counter()

for i in range(len(Row_list_positive)):
        for word in Row_list_positive[i].lower().split(" "):
        positive_counts[word]+=1
        total_counts[word]+=1

for i in range(len(Row_list_negative)):
```

```
            for word in Row_list_negative[i].lower().split(" "):

            negative_counts[word]+=1

            total_counts[word]+=1
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize = (8,8))

negative1.sort_values(by = 'count').plot.barh(x = 'words', y = 'count', ax = ax, color ='purple')

ax.set_title('Most Common Words in Negative Tweets')


fig, ax = plt.subplots(figsize = (8,8))

positive1.sort_values(by = 'count').plot.barh(x = 'words', y = 'count', ax = ax, color ='purple')

ax.set_title('Most Common Words in Positive Tweets')
```

Developing the Misinformation Classifier Model

```
#to clean the data
import re
def  clean_text(df, text_field):
   df[text_field] = df[text_field].str.lower()
   df[text_field] = df[text_field].apply(lambda elem: re.sub(r"(@[A-Za-z0-9]+)|([^0-9A-Za-z
\t])|(\w+:\/\/\S+)|^rt|http.+?", "", elem))
   return df
#df1 is the dataframe containing the labelled tweets which have a column containing the tweets and another
containing the labels.

df1 = clean_text(df, 'tweet')
X, y = df1.tweet, df1.classification

#Another round of cleaning, the tokenization and lemmatization
documents = []

from nltk.stem import WordNetLemmatizer

stemmer = WordNetLemmatizer()

for sen in range(0, len(X)):
   # Remove all the special characters
   document = re.sub(r'\W', ' ', str(X[sen]))
```

```
    # remove all single characters
    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

    # Remove single characters from the start
    document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

    # Substituting multiple spaces with single space
    document = re.sub(r'\s+', ' ', document, flags=re.I)

    # Removing prefixed 'b'
    document = re.sub(r'^b\s+', '', document)

    # Converting to Lowercase
    document = document.lower()

    # Lemmatization
    document = document.split()

    document = [stemmer.lemmatize(word) for word in document]
    document = ' '.join(document)

    documents.append(document)
```

#Vectorizing the data

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=1500, min_df=5, max_df=0.7,
stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(documents).toarray()
```

#Spliting into Test and Train Data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

#RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=1000, random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

#Gaussian Naive Bayes model

```
from sklearn.naive_bayes import GaussianNB

#Create a Gaussian Classifier
gnb = GaussianNB()

#Train the model using the training sets
gnb.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = gnb.predict(X_test)
```

```
#Printing Confusion Matrix for each model
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

Applying the Misinformation Classifier Model

```
#All1 is the dataset of the 75k tweets gathered

df2 = clean_text(All1, 'text')

X_new = df2.text


documents_new = []


from nltk.stem import WordNetLemmatizer


stemmer = WordNetLemmatizer()

for sen in range(0, len(X_new)):

    # Remove all the special characters

    document = re.sub(r'\W', ' ', str(X_new[sen]))

      # remove all single characters

    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

    # Remove single characters from the start

    document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

    # Substituting multiple spaces with single space

    document = re.sub(r'\s+', ' ', document, flags=re.I)
```

```
    # Removing prefixed 'b'
    document = re.sub(r'^b\s+', '', document)

    # Converting to Lowercase
    document = document.lower()
    # Lemmatization
    document = document.split()

    document = [stemmer.lemmatize(word) for word in document]
    document = ' '.join(document)
    documents_new.append(document)
vectorizer1 = CountVectorizer(max_features=56, min_df=5, max_df=0.7, stop_words=stopwords.words('english'))
X_final = vectorizer1.fit_transform(documents_new).toarray()
y_pred_1 = classifier.predict(X_final)
df_predictions=pd.DataFrame(y_pred_1, columns=['predictions'])
df_predictions
```

Plotting Results

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.DataFrame({"Week of the Year 2020": ["Week 43", "Week 44", "Week 45", "Week 46", "Week 47"],
                   "No of Misinformation Tweets": [13769, 13685, 13473, 13512, 13475], "No of Negative Tweets": [7539, 7619, 7846, 8021, 7755],
                   "Total Positive Lab Results": [2611246, 2728875, 3032328, 3356170, 3069436]})
ax1 = sns.set_style(style=None, rc=None )
```

```
import seaborn as sns

import matplotlib.pyplot as plt

sns.set(font_scale=1.5)

fig, ax1 = plt.subplots(1,1,figsize=(12,6))

sns.lineplot(x="Week of the Year 2020", y= 'No of Misinformation Tweets', data = df, marker='^', sort = False, ax=ax1, color="black")

sns.lineplot(x="Week of the Year 2020", y= 'No of Negative Tweets', data = df, marker='o', sort = False, ax=ax1, color="red")

ax2 = ax1.twinx()

sns.barplot(x="Week of the Year 2020", y="Total Positive Lab Results", data=df, alpha=0.5, ax=ax2, color="salmon", saturation=.5 )
```

Appendix for Aim 4

```
# Library's
import json
import sys
import collections
import matplotlib
import warnings
import pandas as pd
from pandas import HDFStore
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt; plt.rcdefaults()
import nltk
import string
import re
#pip install tabulate
from tabulate import tabulate
import seaborn as sns
warnings.filterwarnings("ignore")
from IPython.display import display
pd.options.mode.chained_assignment = None
from sklearn.feature_extraction.text import TfidfVectorizer
pd.set_option('display.max_colwidth', -1)
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt
from PIL import Image
from math import sin, cos, sqrt, atan2, radians

# In[2]:
tweet = pd.read_csv('E:\RESEARCH\FDOT_NA_COVID\Data\FDOT_NA_during_covid.csv',
error_bad_lines=False, warn_bad_lines=False, engine='python', index_col= False, sep='|')
# In[5]:
tweet.columns
# In[4]:
tweet.drop(tweet.columns[10], axis=1, inplace = True)
# In[10]:
tweet.to_pickle("tweet_during")
# In[ ]:
tweet_US=tweet[tweet.Country  == "US"]
```

```
# In[36]:
####
tweet_US.to_pickle("tweet_US_before")
# In[3]:
#tweet_US = pd.read_pickle("tweet_US_before")
# In[4]:
tweet_US.shape
# In[5]:
tweet_US.User_ID.nunique()
# In[6]:
tweet_US=tweet_US[['Time', 'User_ID', 'bounding_box']]
# In[8]:
def find_box_center(row):
    try:
        row_ = eval(row)
        lst_of_coords = [item for sublist in row_ for item in sublist]
        longitude = [radians(p[0]) for p in lst_of_coords]
        latitude = [radians(p[1]) for p in lst_of_coords]
        return (sum(latitude) / float(len(latitude)), sum(longitude) / float(len(longitude)))
    except:
        return None
tweet_US['center_box'] = tweet_US['bounding_box'].apply(lambda row: find_box_center(row))
tweet_US[['center_box_lat', 'center_box_lon']] = pd.DataFrame(tweet_US.center_box.tolist(),
index=tweet_US.index)
# In[11]:
tweet_US.drop(['bounding_box'], axis=1, inplace= True)
# In[15]:
tweet_US['Date']=pd.DatetimeIndex(tweet_US['Time']).date
tweet_US['Day_name']=pd.DatetimeIndex(tweet_US['Time']).day_name()
tweet_US['Hour']=pd.DatetimeIndex(tweet_US['Time']).hour
tweet_US['Mins']=pd.DatetimeIndex(tweet_US['Time']).minute
tweet_US['Sec']=pd.DatetimeIndex(tweet_US['Time']).second
# In[17]:
tweet_US['Date']=pd.to_datetime(tweet_US['Date'])
# In[18]:
tweet_US.drop(['Time'], axis=1, inplace= True)
# In[21]:
tweet_US['time_sec'] = tweet_US['Hour']*3600+tweet_US['Mins']*60+tweet_US['Sec']
# In[30]:
tweet_US
# In[23]:
tweet_US.Date.value_counts().sort_index()
# In[24]:
```

```
day_name = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
tweet_US.Day_name.value_counts().reindex(day_name)
# In[25]:
tweet_US['tweet_count'] = tweet_US.groupby(['Date','User_ID'])['User_ID'].transform('count')
# In[27]:
tweet_US['lat1']=tweet_US.center_box_lat
tweet_US['lon1']=tweet_US.center_box_lon
# In[28]:
tweet_US=tweet_US.assign(lat2=tweet_US.groupby(['Date','User_ID']).center_box_lat.shift(-1),
lon2=tweet_US.groupby(['Date','User_ID']).center_box_lon.shift(-
1),time_sec_=tweet_US.groupby(['Date','User_ID']).time_sec.shift(-1))
# In[29]:
tweet_US["Time_diff"]=tweet_US.time_sec_-tweet_US.time_sec
tweet_US["delta_lon"]=tweet_US.lon2-tweet_US.lon1
tweet_US["delta_lat"]=tweet_US.lat2-tweet_US.lat1
# In[32]:
def a_value(row):

    lat1 = row['lat1']
    lat2 = row['lat2']
    dlon = row['delta_lon']
    dlat = row['delta_lat']

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    return (a)
def dist_value(row):
    a = row['a_conse']
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    d= round(6378*c,9)
    return (d)

# In[33]:
tweet_US['a_conse'] = tweet_US.apply (lambda row: a_value(row), axis=1)
# In[34]:
tweet_US['conse_dist (km)'] = tweet_US.apply (lambda row: dist_value(row), axis=1)
# In[37]:
tweet_US.drop(['center_box', 'Hour', 'Mins', 'Sec', 'a_conse', 'lat1', 'lon1','lat2','lon2' ], axis=1, inplace=
True)
# In[40]:
tweet_US.drop(['delta_lon', 'delta_lat'], axis=1, inplace= True)
# In[41]:
tweet_US
# In[45]:
```

```
####
tweet_US.to_pickle("tweet_US_before_reduced")
# In[43]:
tweet_US['tweeting_speed (m/s)']= tweet_US['conse_dist (km)']*1000/tweet_US["Time_diff"]
# In[44]:
unusual_ID=tweet_US[tweet_US['tweeting_speed (m/s)'] > 240].User_ID.tolist()
# In[46]:
df=tweet_US.drop(tweet_US[tweet_US['User_ID'].isin(unusual_ID)].index)
# In[64]:
###
df.to_pickle("tweet_US_before_reduced_filtered")
# In[50]:
df.drop(['time_sec', 'time_sec_', 'Time_diff'], axis=1, inplace= True)
# In[81]:
df
# In[53]:
df.shape
# In[54]:
df.User_ID.nunique()
# In[55]:
df.User_ID.value_counts()
# In[56]:
df['centroid_lat'] = df.groupby(['Date','User_ID'])['center_box_lat'].transform('mean')
df['centroid_lon'] = df.groupby(['Date','User_ID'])['center_box_lon'].transform('mean')
# In[57]:
df['dlat'] = df['center_box_lat']-df['centroid_lat']
df['dlon'] = df['center_box_lon']-df['centroid_lon']
# In[58]:
def a_value_(row):

    lat1 = row['center_box_lat']
    lat2 = row['centroid_lat']
    dlon = row['dlon']
    dlat = row['dlat']

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    return (a)
def dist_value_(row):
    a = row['a']
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    d= round(6378*c,9)
    return (d)
# In[59]:
```

```
df['a'] = df.apply (lambda row: a_value_(row), axis=1)
# In[62]:
df['rel_dist (km)'] = df.apply (lambda row: dist_value_(row), axis=1)
# In[65]:
df.drop(['dlat', 'dlon', 'a'], axis=1, inplace= True)
# In[72]:
###
df.to_pickle("tweet_US_before_reduced_filtered_reduced")
# In[82]:
df.groupby('Date')['rel_dist (km)'].mean()
# In[69]:
import math as m
def radius_of_gyration(row):
    dist=row['rel_dist (km)']
    n= row['tweet_count']
    rg=sqrt(dist**2/n)
    return (rg)
# In[70]:
df["rg"]=df.apply (lambda row: radius_of_gyration(row), axis=1)
# In[71]:
df.groupby('Date')['rg'].mean()
# In[90]:
df.groupby('Date')['rg'].std()
# In[91]:
df.groupby('Date')['conse_dist (km)'].mean()
# In[87]:
df3=df[df['rg'] > 1]['rg'].value_counts()/df[df['rg'] > 1]['rg'].count()
df3=df3.sort_index().rename_axis('rg').reset_index(name='P(rg)')
# In[88]:
plt.scatter(df3['rg'],df3['P(rg)'],marker='o', facecolors='none', edgecolors='b')
plt.axis([1, 1000, 0.000001, 1])
plt.yscale("log")
plt.xscale("log")
plt.show()
# In[85]:
df4=df[df['conse_dist (km)'] > 1]['conse_dist (km)'].value_counts()/df[df['conse_dist (km)'] >
1]['conse_dist (km)'].count()
df4=df4.sort_index().rename_axis('d').reset_index(name='P(d)')
plt.scatter(df4['d'],df4['P(d)'],marker='o', facecolors='none', edgecolors='r')
plt.axis([1, 1000, 0.0001, 1])
plt.yscale("log")
plt.xscale("log")
plt.show()
```

Reference

Noulas, Anastasios, Salvatore Scellato, Renaud Lambiotte, Massimiliano Pontil, and Cecilia Mascolo. 2012. "A tale of many cities: universal patterns in human urban mobility." PloS one 7, no. 5: e37027. https://doi.org/10.1371/journal.pone.0037027.

Balcan, Duygu, Vittoria Colizza, Bruno Gonçalves, Hao Hu, José J. Ramasco, and Alessandro Vespignani. 2009. "Multiscale mobility networks and the spatial spreading of infectious diseases." Proceedings of the National Academy of Sciences 106, no. 51: 21484-21489. https://doi.org/10.1073/pnas.0906910106.

Wang, Pu, Marta C. González, César A. Hidalgo, and Albert-László Barabási. 2009. "Understanding the spreading patterns of mobile phone viruses." Science 324, no. 5930: 1071-1076. DOI: 10.1126/science.1167053.

Tizzoni, Michele, Paolo Bajardi, Adeline Decuyper, Guillaume Kon Kam King, Christian M. Schneider, Vincent Blondel, Zbigniew Smoreda, Marta C. González, and Vittoria Colizza. 2014. "On the use of human mobility proxies for modeling epidemics." PLoS Comput Biol 10, no. 7: e1003716. https://doi.org/10.1371/journal.pcbi.1003716.

Wilson, Tom, and Martin Bell. 2004"Comparative empirical evaluations of internal migration models in subnational population projections." Journal of Population Research 21, no. 2: 127. https://www.jstor.org/stable/41110801.