MSBD 5003 - Big Data Technology

COVID-19 TWEETS ANALYSIS IN U.S

GROUP 8
CHAN, Yiu Chung
LAM, Chun Ting Jeff
LI, Yilin
WONG, Ka Wing

NOVEMBER 29, 2021

HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Table of Contents

Task Statement

The COVID-19 pandemic has in many ways affected people's lives worldwide. Under the new norms of social distancing, the popularity and importance of online social networks have reached an unprecedented level. Social media becomes a rich digital channel for people to express their emotions, share thoughts, and support each other. Twitter, like other major networks, has seen a record number of users during the pandemic. Given its outstanding speed in news spread, vast user base, and the micro-blog nature, Twitter constantly generates concise posts during the pandemic, making it the perfect data source.

Our team intends to leverage the high-volume Twitter data to understand people's emotions and thoughts in the United States throughout the COVID-19 period. We hope to provide valuable insights into the effect of the outbreak on the general public and in turn offer useful information to help world leaders and organizations navigate through this crisis. To this end, our objectives include the following,

- Analyze public response during pandemic
- Explore Twitter data related to coronavirus
- Create a live streaming of COVID tweets analysis

The first objective is to analyze the trend of public emotions and sentiments about COVID-19 on Twitter. Due to the limit in search quota and computational power, we choose tweets from 27 January 2020 to 1 September 2021. Daily data such as new cases, death and test positive rate are also combined with our sentiment analysis results to gain more understanding of the reasoning behind.

The second objective is to perform text mining of the Twitter message. For example, words that appear most often in the Tweets could provide insight on what is driving the discussion on Twitter. Topic modeling is also used for classifying Tweets into different topics based on their similarity and for figuring out emotions and sentiments associated with different topics.

The third objective is to perform real-time analysis on tweets. Based on the historical Twitter data with sentiments labeled in the first objective, we would train a sentiment analysis model to classify the incoming tweets. Eventually, the real time tweets data with sentiment information will be displayed on a dashboard.

Dataset Description

Twitter Data

The first dataset contains all the tweets about COVID-19 in the U.S. from 27 January 2020 to 1 September 2021 with their sentiment labelled. This dataset is created by combining data from the COVID-19 Data Repository[1] and the Twitter API.

Column	Data Type	Description
Tweet ID	String	Unique ID for each tweet
Timestamp	DateTime	When the tweet is created
Text	String	The full tweet content
Retweet Count	Int	Number of retweet(s)
Quote Count	Int	Number of quote(s)
Like Count	Int	Number of like(s)
Reply Count	Int	Number of reply(s)
Sentiment	String	Positive, Negative, Neutral

Table 1: Labeled Twitter data including user interactions

Since there are limits in both the search tweets quota in Twitter API and our computing resources, we exploit the COVID-19 Data Repository[1] to build our dataset. The repository[1] contains a full set of tweet IDS based on keywords “corona”, “wuhan”, “nCov” and “covid” during pandemic . Also, the repository[1] has already classified the sentiments of those tweets in 5 classes, “very negative”, “negative”, “neutral or mixed”, “positive”, “very positive”.

Since the repository[1] only contains the tweets IDs and creation date, we need to use the Twitter API to get detailed information about the tweets, including full text, retweet count, quote count, favorite count and reply count. Also, in order to reduce complexity, the sentiments of the tweets are only classified into 3 classes, “negative”, “neutral” and “positive”.

Overall, there are 54380875 records.

tweet_id	tweet_timestamp	text	retweet_count	quote_count	like_count	reply_count	sentiment	emotion
1221982095444100000	2020-01-27 18-23-29	Singapore-listed Sasseur REIT shuts China malls as Wuhan	0	0	0	0	negative	fear
1222053147385810000	2020-01-27 23-05-49	@C_Barraud Here's a message from a supplier of chemicals	0	0	0	0	neutral or mixed	no specific emotion
1222059789116750000	2020-01-27 23-32-13	EXPOSED LEAKS FROM STAFF SAY 90K DEAD #Wu	0	0	0	0	negative	fear
1222070757120970000	2020-01-28 00-15-48	US to expand virus screening at 20 airports for visitors from	1	1	1	0	negative	fear
1222079623825160000	2020-01-28 00-51-02	Chinese Premier Visits Wuhan as Virus Death Toll Rises h	0	0	0	0	negative	fear

Table 2: example tweet detail data including user interactions

Daily Coronavirus Data

The second dataset is the daily coronavirus data in the US from [2]. [2] provides a holistic dataset of COVID statistics over the world, which is a combination of data from various sources such as government departments and research organizations. This dataset is critical for us to understand how the pandemic is developing. We try to analyze trends in the development of covid and its association with the tweets.

Column	Data Type	Description
Date	Date	Report date
Total Cases	Integer	No. of confirmed case
New Cases	Integer	No. of new confirmed case
Total Deaths	Integer	No. of deaths

New Deaths	Integer	No. of new deaths
Hospital Patients	Integer	No. of hospital patients
People Vaccinated	Integer	No. of people vaccinated (include partially vaccinated)
New Vaccinations	Integer	No. of people new vaccinations

Table 3: Daily coronavirus data

Technologies

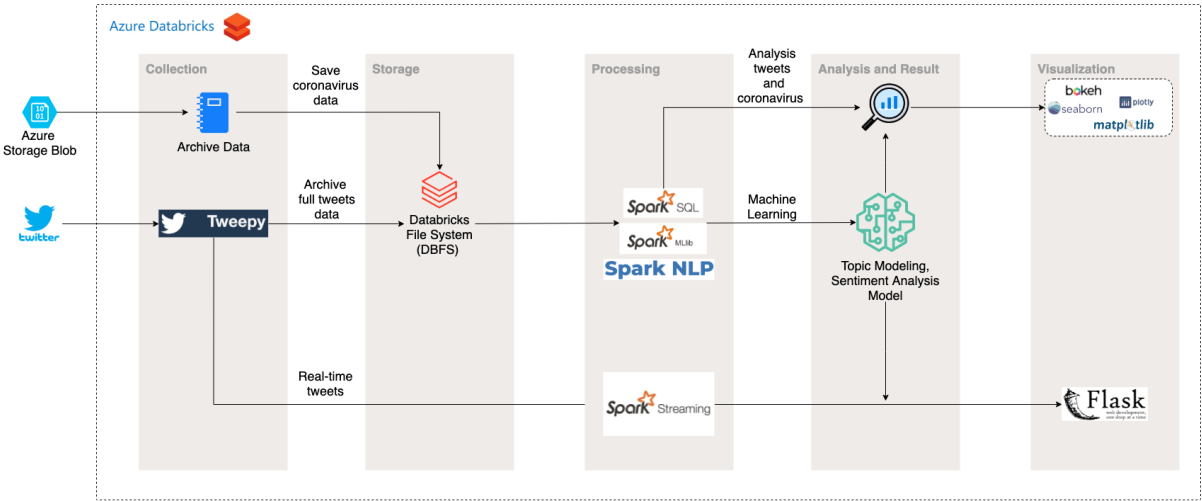


Figure 1: Technologies used for this project

Data Collection	Tweepy
Data Storage	Azure Storage Blob, Databrick File System
Data Utility & Preprocessing	NLTK, pandas, numpy, Spark RDD, Spark SQL
Data Exploration & Analysis	Spark SQL, Spark MLlib, Spark NLP, Spark Streaming
Data Visualization	Flask, bokeh, seaborn

Table 4: Technologies by category

Most parts of our architecture are built on top of Databricks. Databricks is a well-managed platform for running Apache Spark. It is user-friendly but efficient. Based on it, we do not need to handle those complex cluster management concepts or perform tedious maintenance tasks while we can take full advantage of Spark.

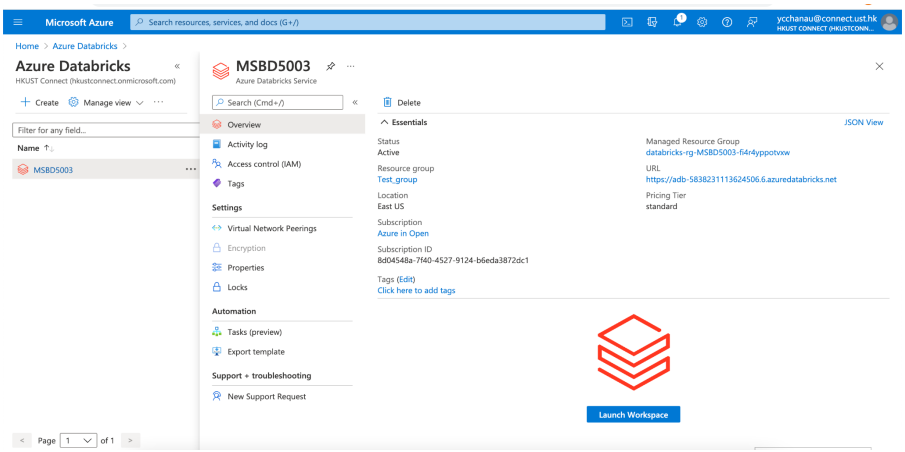


Figure 2: Azure Databricks

Data Collection

Tweepy

Tweepy is a python library which wraps the Twitter API in an easy-to-use way. We use it to access twitter data. **Client.get_tweets** will be used to archive full data based on tweet ids. **tweepy.Stream** will be used to listen to the stream data from Twitter.

```
api = tweepy.API(auth)
client = tweepy.Client(bearer_token=bearer_token,
                       consumer_key=consumer_key,
                       consumer_secret=consumer_secret,
                       access_token=access_token,
                       access_token_secret=access_token_secret)

def tweet2tuple_v2(t):
    pm = t['public_metrics']
    return (str(t["id"]), t["text"].replace("\n"," "), t["lang"], pm['retweet_count'],
            pm['quote_count'], pm['like_count'], pm['reply_count'])


def query_tweets_v2(row):
    ids = row["ids"]
    response = client.get_tweets(ids=ids, tweet_fields=["text", "lang", "public_metrics"])
    tweets = response.data
    for tweet in tweets:
        yield tweet2tuple_v2(tweet)
```

Figure 3: Get_Detailed_Tweets_Info.html

Data Storage

There are two parts of data storage --Azure Storage Blob and Databrick File System(DBFS). Storage Blob is used to store large raw data. DBFS will mainly be used to store the processed data, so that we can access them easily while performing analysis.

Storage Blob

We choose to store our large raw data with Storage Blob because it is cheap and easy to set up. Compared with other large data storage methods like creating a HDFS cluster on Azure or the Azure Data Lake Storage Gen2, the reading and writing cost of Storage Blob is the lowest and it only takes a few steps to set up. Also, under the geo-redundant storage our data is copied to a second region. We can recover the data easily after a disaster which ensures reliability. In addition, we can easily archive the data from Storage Blob to DBFS with a few lines of code.

Cmd 1

```
storage_account_name = "yc5003"
storage_account_access_key = "KhziZq7zXE1KxpmVzMI9stSABRVWFVXvOgsiJXWCLPdAn0ZSeJb20xFLpb81DE72Pe4l9MLJS2nV/estArLmoA=="
```

Cmd 2

```
directory_name =  "tweetid_userid_keyword_sentiments_emotions_United_States.csv"
container_name = "covid"

file_location = f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net/{directory_name}"
file_type = "csv"

table_name = "tweets_us"

file_location
```

Cmd 3

```
spark.conf.set(
  "fs.azure.account.key."+storage_account_name+".blob.core.windows.net",
  storage_account_access_key)
```

Cmd 4

```
df = spark.read.format(file_type).option("header", "true").load(file_location)
df.printSchema()
```

Figure 4: Azure Blob Storage Import Original Dataset.html

DBFS

DBFS is a distributed file system on the Databrick clusters. It will be mounted to the Databrick workplace once we start the cluster. It is intuitive to use DBFS instead of setting up an additional HDFS cluster since we will carry out our project under Databrick. DBFS covers most of the functionalities of a distributed file system and can be seamlessly accessed by the Spark API. Our tables and trained machine learning models are also saved here.

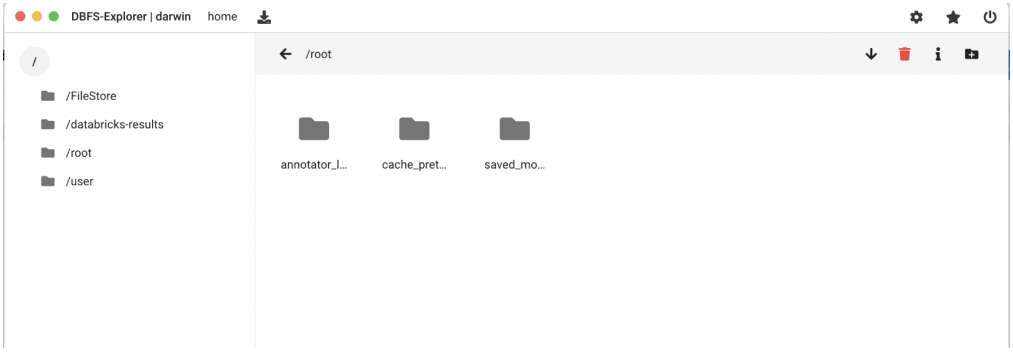


Figure 5: DBFS

Data Utility & Preprocessing

Multiple tools will be used for preprocessing and we will mainly introduce NLTK, Spark SQL.

Spark SQL

Spark SQL is a module of Spark which is optimized for structured data processing. It provides a user-friendly way for us to process our data without worrying about the underlying details. It can also help optimize our commands based on several techniques. We mainly use it for transformation, filtering, grouping, ordering and removal of duplicate records during pre-processing.

Spark NLP

Spark NLP is a state-of-the-art, mission-critical and enterprise-grade Natural Language Processing library built atop Spark. We choose Spark NLP instead of NLTK since the former is optimized for distributed environments and has provided a lot of well-defined apis to pre-process the text data before performing machine learning tasks . Based on it , pre-processed steps such as tokenization, lemmatization, and removal of stop words, can be carried out easily and efficiently.

Steps To Clean The Data

- Since every tweet has an unique tweets ID, we first remove those duplicated tweets.

- When collecting the tweet details, we notice that some tweets are already unavailable because the users may have deleted their posts or Twitter perceived those tweets as scam and fake news and have deleted them. We also remove these tweets from our dataset.

Eventually, 46334395 tweets remained.

- Regarding the text in tweets, we first convert all the text to lowercase. Next, rrls, other users mentioned with the symbol "@", hashtags with the symbol "#", numbers, emojis, and symbols are removed. There are also some censored words in text and some people like to repeat a specific character in a word. These special cases are also handled.

```
def remove_symbols(tweet):
    tweet = re.sub(r'[\"\\\'\(\)\-\[\]\,\.\?\!\$\%\:\*\>\<\;\{\}\+\|\=]', ' ', tweet)
    return tweet

def remove_extra_spaces(tweet):
    tweet = re.sub(r'\s+', ' ', tweet)
    return tweet

arr = [remove_links, remove_users, remove_hashtag,
       remove_number, remove_emoji, handle_special_wrods,
       remove_repeated_characters, remove_symbols, remove_extra_spaces]
def _normalize(tweet):
    tweet = tweet.lower()
    for f in arr:
        tweet = f(tweet)
    return tweet
```

Figure 6: Preprocessing.html

- As mentioned before, we eventually carried out tokenization, lemmatization, and removal of stop words, to transform the text to tokens for the machine learning tasks in the next part.

```
tokenizer = Tokenizer() \
    .setInputCols(['document']) \
    .setOutputCol('token')

# spell_checker = NorvigSweetingModel.pretrained() \
#   .setInputCols(["token"]) \
#   .setOutputCol("checked_token")

normalizer = Normalizer() \
     .setInputCols(["token"]) \
     .setOutputCol('normalized') \
     .setLowercase(True)

lemmatizer = LemmatizerModel.pretrained() \
     .setInputCols(['normalized']) \
     .setOutputCol('lemmatized')

eng_stopwords = stopwords.words('english')
stopwords_cleaner = StopWordsCleaner() \
     .setInputCols(['lemmatized']) \
     .setOutputCol('no_stop_lemmatized') \
     .setStopWords(eng_stopwords)

finisher = Finisher() \
    .setInputCols("no_stop_lemmatized") \
    .setOutputCols("output")

stages = [document_assembler,
                tokenizer,
               #  spell_checker,
                normalizer,
                lemmatizer,
                stopwords_cleaner,
                finisher
          ]

pipeline = Pipeline() \
     .setStages(stages)
empty_df = spark.createDataFrame([['']]).toDF("text")
pipelineModel = pipeline.fit(empty_df)
```

Figure 7: Preprocessing.html

Data Exploration & Analysis

Spark MLlib

Spark MlLib is a powerful module of Spark, which allows us to perform machine learning on large-volume data in a practical and scalable way. It conforms to Spark API and can be incorporated with Numpy in python. As we need to handle millions of tweets, we decide to use it instead of sklearn or other machine learning libraries which mainly focus on small datasets. We mainly use it for two purposes. First, we use LDA from the clustering algorithms in MLlib for topic modeling of tweets. Second, we use different classification algorithms to predict the sentiments of tweets.

To perform machine learning, we first use the traditional bag-of-words method, vectorizing the text with count vectorizer and tf-idf.

```
Cmd 13
string_indexer = StringIndexer(inputCol="label", outputCol='label_indexed')

vocabSize=1000
minDF=5
count_vectorizer = CountVectorizer(
    inputCol='tokens',
    outputCol='raw_features',
    minDF=minDF,
    vocabSize=vocabSize )

idf = IDF(inputCol="raw_features", outputCol="features", minDocFreq=minDF)
```

```
Cmd 14
pre_process_pipeline = Pipeline(stages=
                    [
                     string_indexer,
                     count_vectorizer,
                     idf
                     ])
pre_process_model = pre_process_pipeline.fit(df)
```

Figure 8: Train Sentiment Analysis.html

To analyze the tweets more deeply, we want to figure out the underlying sub-topics of these COVID-related tweets. Therefore, we make use of topic modeling, which is an unsupervised machine learning technique that can automatically clustering documents based on words within them.

```
from pyspark.ml.clustering import LDA
num_topics = 10
max_iter = 40
lda = LDA(k=num_topics,
          maxIter=max_iter,
          featuresCol='features')
```

Figure 9: Topic Modelling.html

To classify the sentiment of incoming tweets, we exploit three supervised machine-learning techniques, Naive Bayesian, Logistic Regression and Multi-Layer Perceptron.

```
classifier_1 = NaiveBayes(
    featuresCol='features',
    labelCol='label_indexed',
    predictionCol='pred',
)

classifier_2 = LogisticRegression(
    featuresCol='features',
    labelCol='label_indexed',
    predictionCol='pred',
    maxIter=100,
    regParam=0.1,
)

input_size = CountVectorizerModel.getVocabSize()
ouput_size = len(StringIndexerModel.labels)

classifier_3 = MultilayerPerceptronClassifier(
    featuresCol='features',
    labelCol='label_indexed',
    predictionCol='pred',
    maxIter=15,
    layers=[input_size, input_size//10, ouput_size]
)
```

Figure 10: Train Sentiment Analysis.html

Spark Streaming

Spark Streaming is an extension tool for Spark that allows us to handle large-velocity and large-volume streaming data with high-throughout, scalability and fault tolerance. High-Level operations like map, filter, reduce can be performed on the incoming live data from various sources such TCP

sockets and Kafla[3]. Eventually, the processed data can be sunk to filesystems, databases, and live dashboards. In our case, we use a live board to display the twitter data based on Spark Streaming. More detail will be discussed in the later sections.

Data Visualization

Flask, bokeh, seaborn

These tools are used for data visualization. In this project, we will choose one of these tools to visualize our final outcome and create a sentiment information dashboard. The main objective of using these tools is to provide a user-friendly interface so as to let users monitor the system easily. These tools also help us get feedback and control parameters in a more effective way.

Result 1: The Public Response on Twitter during Pandemic

* Find the Code in “Analysis_And_Visualize/Task_1”

We counted the number of covid related tweets posted every day, and found an interesting fact: overall the covid topic trend has ups and downs, but locally the topic trend intensity follows a weekly pattern. Sundays are when covid is least discussed, followed by an upward trend during the middle of the week, and then a downward trend until the end of the week.

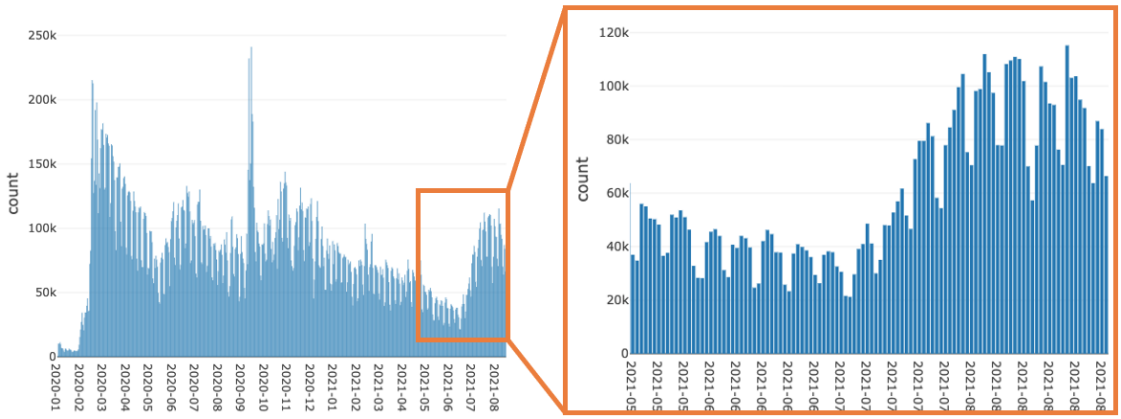
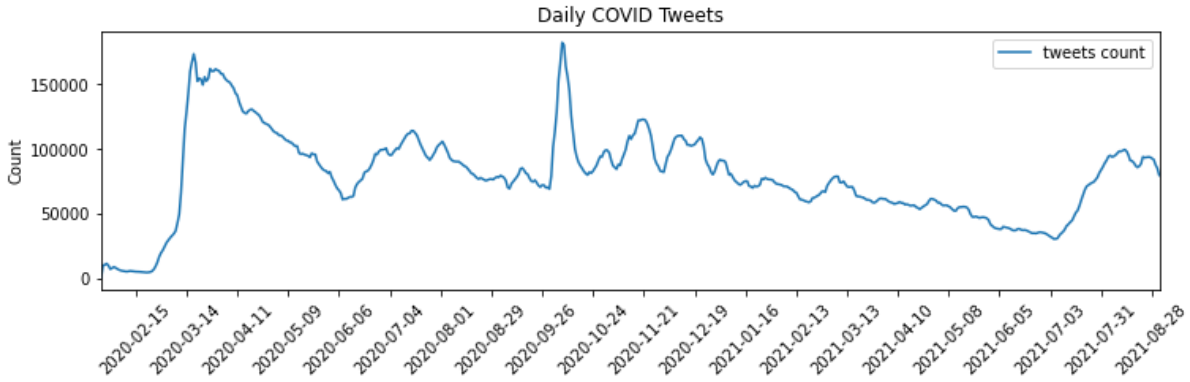
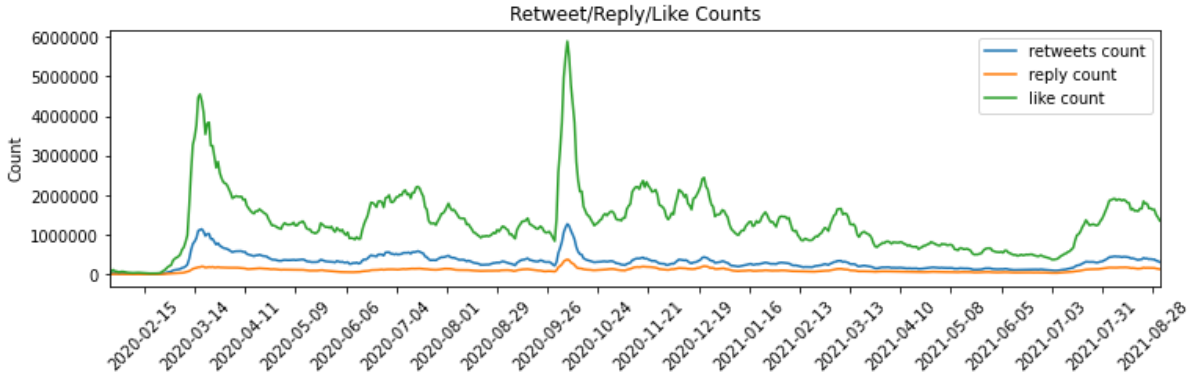
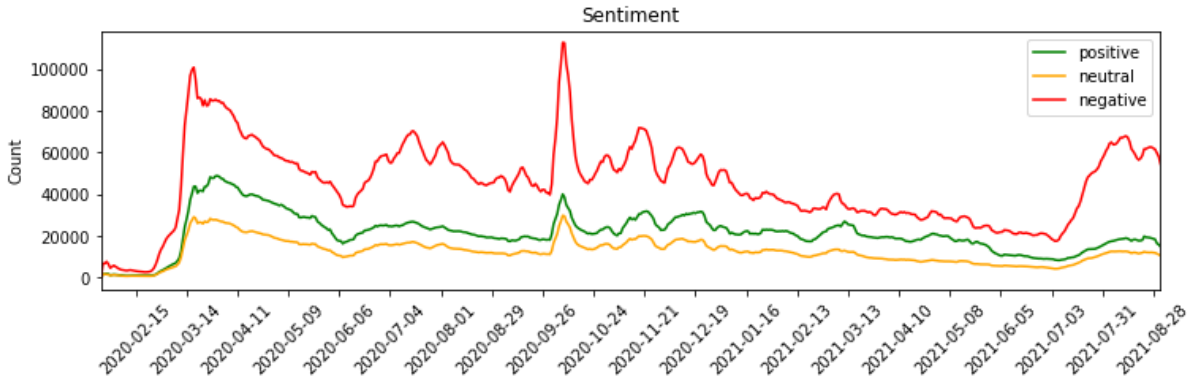


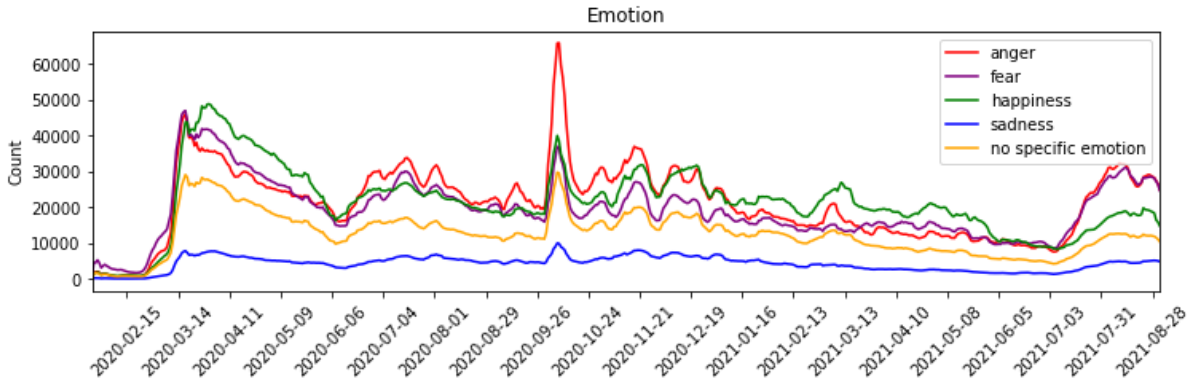
Figure 11: Count of daily COVID-19 tweets

Given this pattern, we applied a 7 day moving average to smooth out the twitter statistics so that the overall trends are more clear. Our findings are as follows.


Daily COVID Tweets
tweets count
Count
150000
100000
50000
0
2020-02-15
2020-03-14
2020-04-11
2020-05-09
2020-06-06
2020-07-04
2020-08-01
2020-08-29
2020-09-26
2020-10-24
2020-11-21
2020-12-19
2021-01-16
2021-02-13
2021-03-13
2021-04-10
2021-05-08
2021-06-05
2021-07-03
2021-07-31
2021-08-28


Retweet/Reply/Like Counts
retweets count
reply count
like count
Count
6000000
5000000
4000000
3000000
2000000
1000000
0
2020-02-15
2020-03-14
2020-04-11
2020-05-09
2020-06-06
2020-07-04
2020-08-01
2020-08-29
2020-09-26
2020-10-24
2020-11-21
2020-12-19
2021-01-16
2021-02-13
2021-03-13
2021-04-10
2021-05-08
2021-06-05
2021-07-03
2021-07-31
2021-08-28


Sentiment
positive
neutral
negative
Count
100000
80000
60000
40000
20000
0
2020-02-15
2020-03-14
2020-04-11
2020-05-09
2020-06-06
2020-07-04
2020-08-01
2020-08-29
2020-09-26
2020-10-24
2020-11-21
2020-12-19
2021-01-16
2021-02-13
2021-03-13
2021-04-10
2021-05-08
2021-06-05
2021-07-03
2021-07-31
2021-08-28


Emotion
anger
fear
happiness
sadness
no specific emotion
Count
60000
50000
40000
30000
20000
10000
0
2020-02-15
2020-03-14
2020-04-11
2020-05-09
2020-06-06
2020-07-04
2020-08-01
2020-08-29
2020-09-26
2020-10-24
2020-11-21
2020-12-19
2021-01-16
2021-02-13
2021-03-13
2021-04-10
2021-05-08
2021-06-05
2021-07-03
2021-07-31
2021-08-28

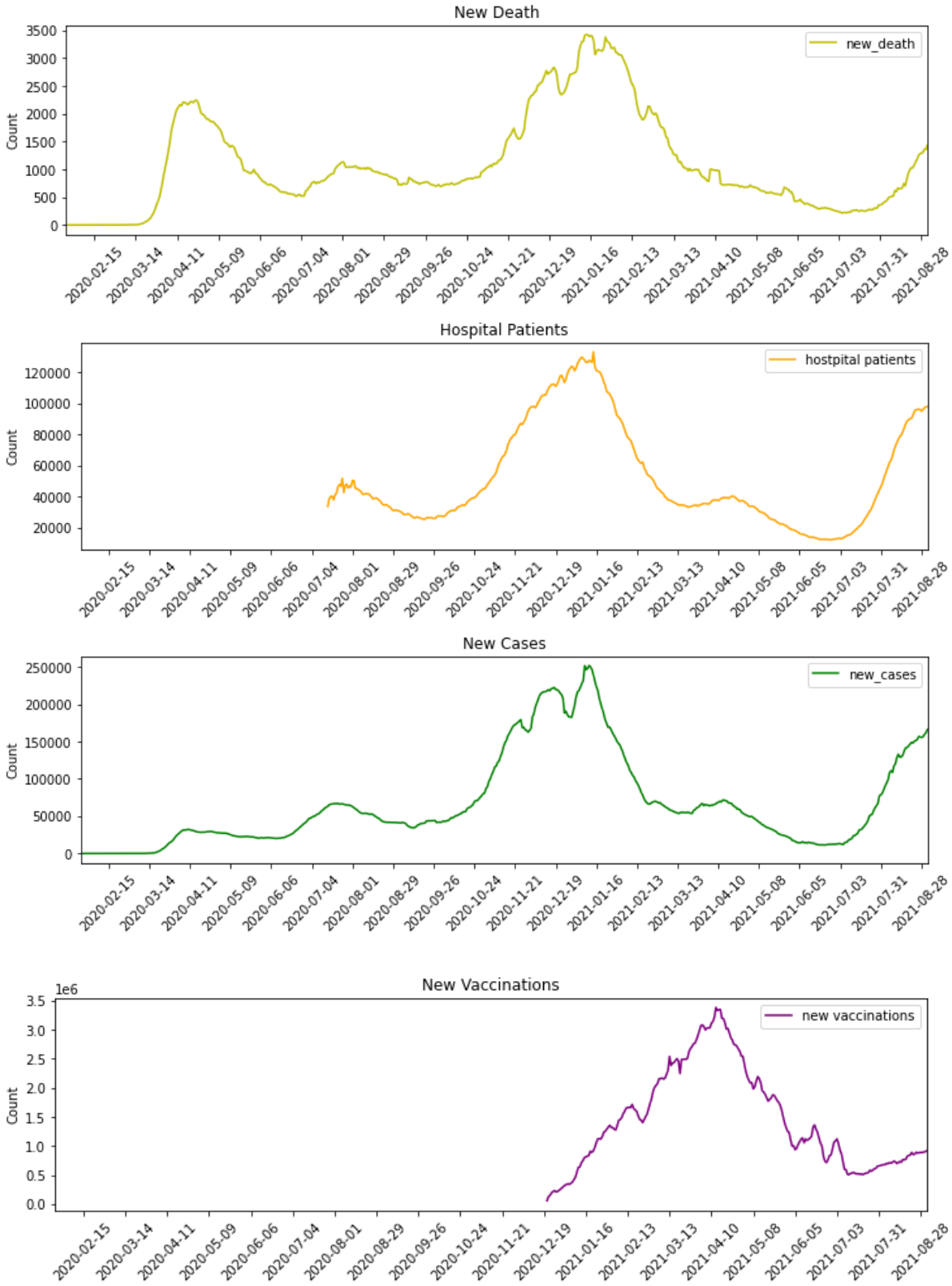


Figure 12: COVID-19 Twitter Metrics

The public's response to COVID on Twitter is influenced by multiple factors.

At the beginning of 2020, the overall number of COVID tweets is closely related to the severity of the pandemic. There was a sharp increase in new deaths from late February to mid March, 2020. And on

March 11th, WHO Declared COVID-19 a Pandemic. 2 days after, the former president Trump declared a National Emergency. The daily number of COVID tweets reached its first peak on the same day. Later on due to stay-at-home orders and other tightened policies, the daily new deaths decreased, and the daily number of tweets on COVID decreased as well.

On October 2, news broke that the then president Trump and First Lady Test Positive for COVID-19, and Trump entered hospital on the same day. This brought the COVID discussion to its second peak. On that day, the likes received by COVID tweets are about 2.5 times the likes of an average day.

In the following 9 months, we cannot observe the close tie between the number of COVID tweets and the pandemic severity any more. There's another increase in daily death numbers from early October, 2020 to mid-January, 2021. However, the number of COVID tweets remained about the same level as before Trump was admitted. We also tried to see if positive efforts such as vaccination had any effect on the public response. However, we weren't able to find such relations.

The second wave that started early July, 2021, again brought COVID to the public's attention. Luckily this time, even though the number of new cases and hospitalization increased sharply, the number of new deaths did not grow as fast.

Result 2: Twitter Data Exploration

* Find the Code in "Analysis_And_Visualize/Task_2"

Distribution of Sentiments and Emotions

We have conducted an analysis of distribution of sentiments and emotions of the tweets during the period from January 2020, which is also the start of the Covid-19 outbreak, up to September 2021.

From the pie chart, we can see that about 60% of the tweets during the period are negative and only about 30% are positive.

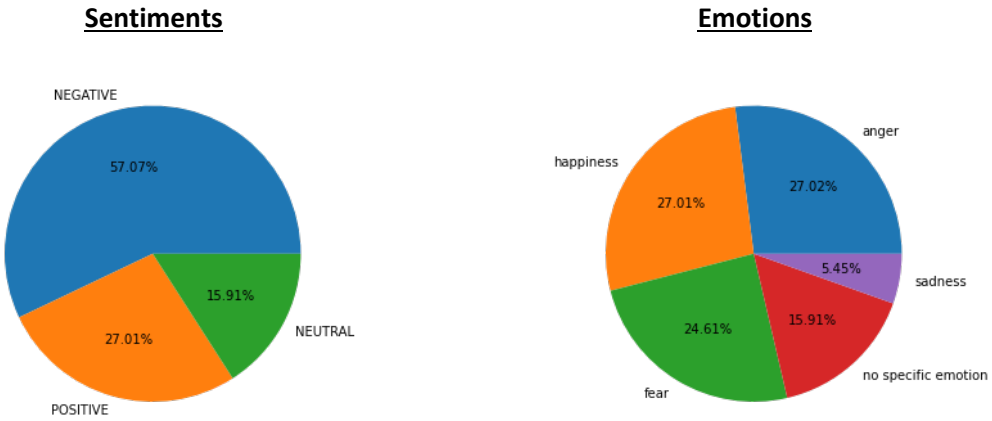


Figure 13: Sentiments and Emotions Distribution

And the majority of the negative tweets are related to the emotion of fear and anger.

The result seems reasonable.

People may feel angry as they are unable to meet their friends or participate in different outdoor activities under the social distancing measures. People also cannot travel to other countries due to the border control.

People may also feel fear as the virus seems to be highly contagious and a high death rate is observed in certain countries.

Word Cloud of Top Words

A word cloud is also generated for the 300 keywords that appear most often in the tweets, with font size proportional to the frequency that they appear.

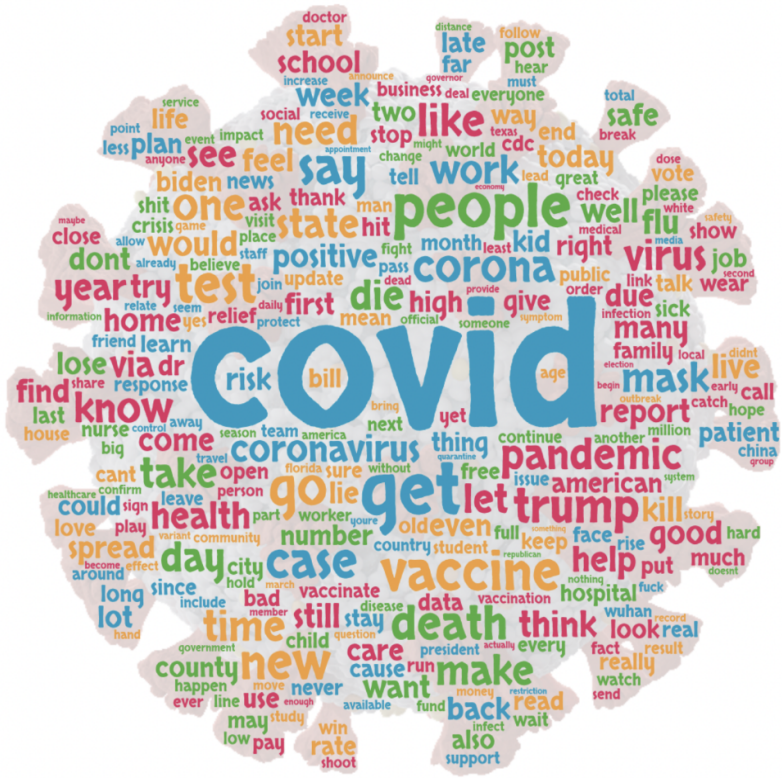


Figure 14: COVID-19 Word Cloud

It comes as no surprise that Covid is the major focus. Other keywords are also highly related to the Covid outbreak as well, such as vaccine, test, mask and case.

Donald Trump also gets quite a lot of attention during the period. The reason may be that the US President Election was conducted in 2020.

Characteristics of Tweets by Emotion

We have tried to discover other characteristics of the tweets as well. For example, whether there would be association between emotion of a tweet and its word count, retweet count and reply count.

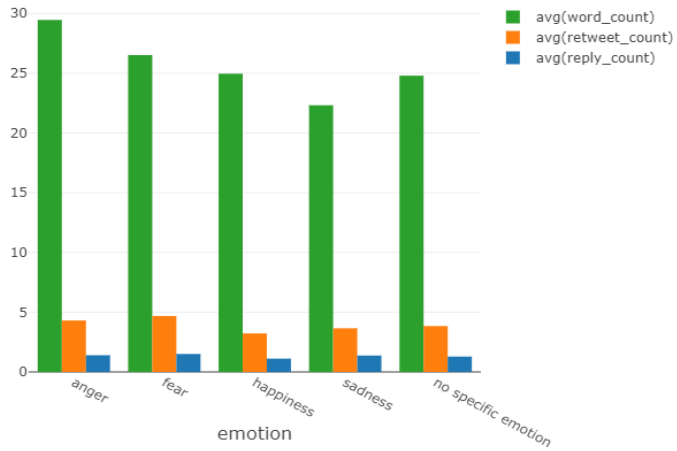


Figure 15: Average Emotions

The difference is not quite obvious for tweets related to different emotions.

Maybe the users who have created a tweet are more influential towards the reply or retweet count than the emotion of the tweet.

Topic Modeling of Tweets

To figure out the latent topics in the tweets, Latent Dirichlet Allocation (LDA) which is a widely-used topic modeling method is adopted. We set the number of clusters to 10 and the feature size to 30000, iterating 40 epoches.

	Topic	Possible Topic	Topic Words
	0	Personal Confirmed Diagnosis	get, corona, people, im, go, know, like, dont, die, think, take, sick, say, flu, test
	1	Media News	test, coronavirus, health, new, state, update, positive, city, county, pandemic, news, via, south, worker, official
	2	Prevention	vaccine, mask, johnson, mandate, pfizer, fda, say, trial, amp, trump, via, super, test, effective, biden
	3	Community Collaboration	pandemic, help, learn, business, impact, amp, health, support, community, crisis, join, need, student, work, relief
	4	Healthcare Provision	people, patient, nurse, get, cause, death, would, trump, amp, condition, die, home, make, hospital, like
	5	Social Distance Policy	mask, bill, distance, wear, social, gt, go, face, relief, get, amp, pay, say, need, vaccine
	6	Pandemics in U.S. / Appointment of Vaccine	life, american, walgreens, people, link, death, die, provider, available, child, appointment,

			number, location, lose, detect
	7	China-U.S. Relationship	trump, virus, wuhan, china, corona, like, biden, say, world, people, call, president, media, go, lab
	8	School Policy for Covid	test, get, school, year, go, game, site, today, time, week, vaccine, first, day, one, post
	9	Index Cases	case, death, report, new, county, state, number, day, confirm, test, total, rate, patient, update, coronavirus

Table 4: Top 15 key

We can see some underlying topics from the top 15 keywords. For example, topic #2 is about the prevention methods and topic #7 is about the virus in WuHan, China and the U.S. government.

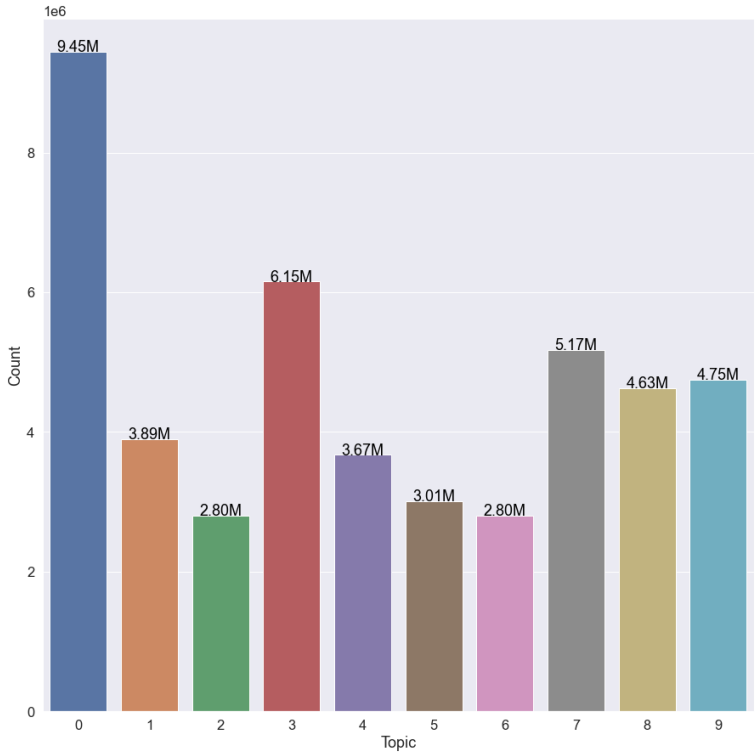


Figure 16: Topic Count

From the table, the top 3 most popular topics are topic #0 , topic #3 and topic #7

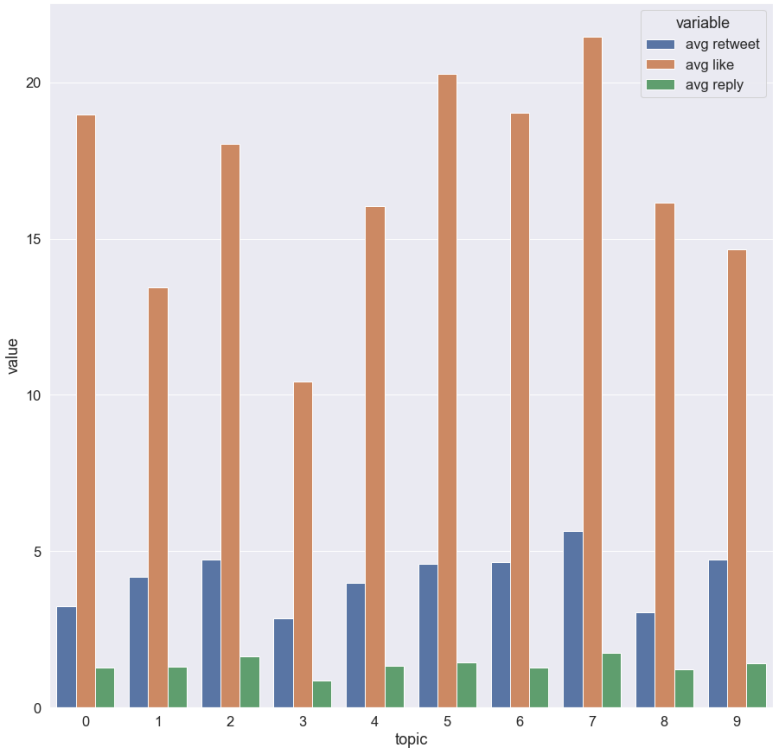


Figure 17: Average retweet, like, reply of each topic

From the table, we can find that topic #7 is the most engaged topic. It has the highest value of average retweet, like and reply. Considering its key words, this topic may be related to conspiracy theory and China–United States relations, thus invoking so many responses.

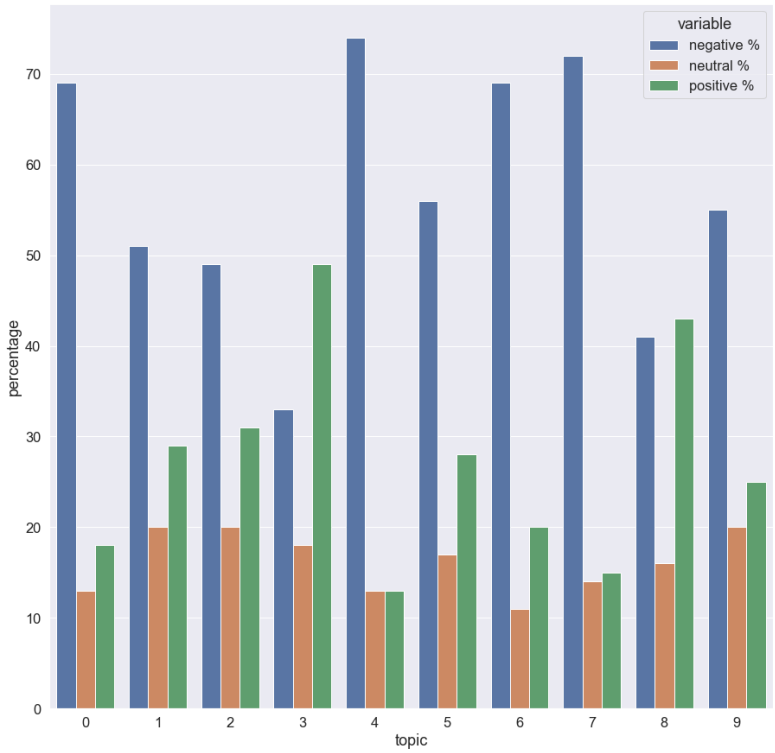


Figure 18: Emotion distribution of each topic

First, we can observe that for most topics, the negative sentiment dominates. However, in topic #3

and topic #8, the positive sentiment becomes predominant. If we look back to the keywords, topic #3 is about community cooperation and topic #8 is about school policy.

Time vs Topic

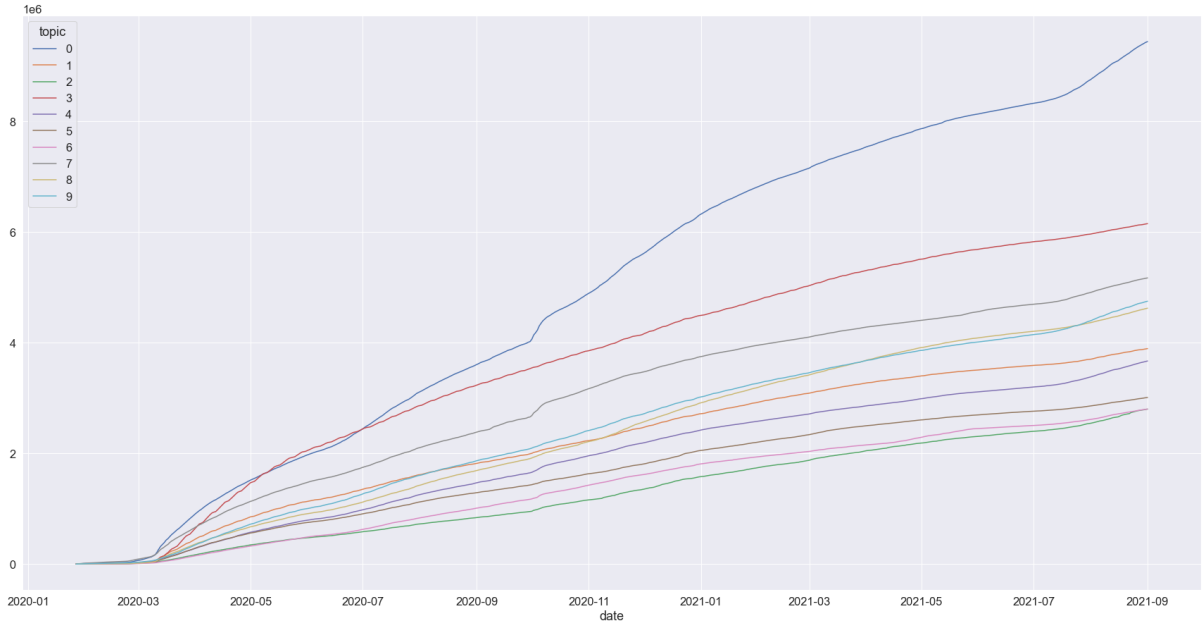


Figure 19: Total number of tweets for each topic

We also produce a short bar chart race video about how the number of tweets of each topic changed from 27/01/2020 to 01/09/2021. From the video, we can clearly see that there are abrupt increases in the total number of topic #0 (Personal Confirmed Diagnosis) at the beginning of March 2020 and October 2020, where are the peaks of the pandemics.

Result 3: Live Tweets Analysis

* Find the Code in “Analysis_And_Visualize/Task_3”

Sentiment Classification of Tweets

To understand the real-time public sentiment regarding COVID, a classifier is required to classify the sentiment of tweets into 3 classes, “negative”, “neutral”, “positive”. Therefore, we have trained different models based on the tweets data collected.

Since the dataset is highly imbalanced, the number of negative and positive tweets is much larger than neutral tweets, therefore we first perform under sampling to produce a balanced dataset for training, and the remaining data is used for validation. As a result, the size of the train set and validation set are 23892550 and 22441845 respectively.

```
def split_train_test(df):

    neu_df = df.filter(col("label")==NEUTRAL)
    pos_df = df.filter(col("label")==POSITIVE)
    neg_df = df.filter(col("label")==NEGATIVE)

    neu_ratio = 0.159
    neu_split_ratio = 0.9
    need_fraction = neu_ratio * neu_split_ratio

    neg_ratio = 0.570
    neg_split_ratio = need_fraction / neg_ratio * 1.35
    pos_ratio = 0.270
    pos_split_ratio = need_fraction / pos_ratio * 1.25

    def foo(split_ratio):
        return [split_ratio, 1-split_ratio]


    neu_train, neu_val = neu_df.randomSplit(foo(neu_split_ratio), seed=100)
#     print(neu_train.count(), neu_val.count())
    neg_train, neg_val = neg_df.randomSplit(foo(neg_split_ratio), seed=100)
#     print(neg_train.count(), neg_val.count())
    pos_train, pos_val = pos_df.randomSplit(foo(pos_split_ratio), seed=100)
#     print(pos_train.count(), pos_val.count())

    train_set = neu_train.union(neg_train).union(pos_train).repartition(NUM_OF_CORES)
    test_set = neu_val.union(neg_val).union(pos_val).repartition(NUM_OF_CORES)

    return train_set, test_set
```

Figure 20: Train Sentiment Analysis.html

We have tried three built-in classification algorithms in SparkMLlib, Naive Bayesian, Logistic Regression and Multi-Layer Perceptron.

Model	Acc	F1-Score
Naive Bayesian	0.67	0.74
Logistic Regression	0.79	0.82
Multi-Layer Perceptron	0.76	0.78

Table 5: Model Accuracy and F1-Score

Naive Bayesian is a simple classifier based on probability methods. Unlike other methods, it doesn't require iterative approximation and thus has a short training time and is highly scalable. Therefore, it is used as a baseline in this task. We set the feature size to 16000 for training. From the table, the accuracy of it is not high, possibly due to its simplicity.

Multinomial logistic regression is a generalized version of logistic regression for multi-class classification. It tries to figure out a model to separate the data linearly. We set the feature size to 16000, iterate 100 epoches with regularization parameter 0.1. In our task, this method achieves the highest accuracy and F1-score, therefore, it is adopted to classify the real-time tweets.
Multilayer Perceptron is a class of artificial neural networks. With a non-linear activation function, it can adapt to any function, thereby classifying linearly-inseparable data. In our model, the input size, hidden layer size, output size is 1000, 100, 3, and sigmoid activation function is adopted. We iterate 15 epochs and it surprises us that the accuracy of it is lower than logistic regression. One possible explanation is that the input size is too small. However, since our computation power is limited, we fail to test a bigger model.

Showing Real-Time Statistics of COVID-related Tweets

There are mainly three components for streaming. First component is the twitter streaming, which acts as a bridge between twitter to spark. The second component spark itself, which will send a request to twitter streaming for tweets data, and process the data. The third component is Flask, which will receive the data from pyspark, and display it as web content.

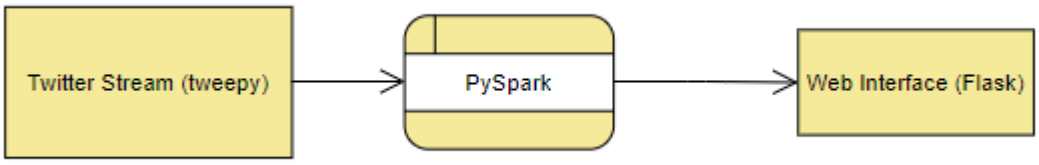


Figure 21: Streaming Components

Twitter Streaming

Tweepy library version 4.0 has been used for receiving data from twitter. An intermediate server script has been implemented once client requests the stream, twitter data will be grabbed from Twitter API, and will send the processed full text data to client when on_data event triggered. Port 5555 will be exposed for request, and tags of 'covid', 'coronavirus', 'cov', 'corona' are used for parameter searching in twitter.

```
def start(self, keyword):
    try:
      # server (local machine) creates listening socket
      s = socket.socket()
      host = "0.0.0.0"
      port = 5555
      s.bind((host, port))
      # server (local machine) listens for connections
      s.listen(4)
      print('Server is listening on (IP: '+str(host)+',Port: '+str(port)+')... ')
      # return the socket and the address on the other side of the connection (client side)
      c_socket, addr = s.accept()
      print("Received request from: " + str(addr))
      # select here the keyword for the tweet data
      self.__sendData(c_socket, keyword)
    except Exception as e:
      print(e)
      s.close()
```

Figure 22: Tweepy Live Stream And Spark Streaming/main.py]

Pyspark Streaming

A batch processing which is triggered per 10 second has been created to collect the twitter data from tweepy. After receiving the plain twitter dataset, it will pass the data to the sentiment analysis model for processing. Three classes will be classified for that twitter data, which are positive, neutral, and negative. For details in classification, please refer to section 3.1. We also collect the most frequently appearing hashtags. We record the data within a window of 1 hour and slide interval of 10 second. After that, the data will be passed to Flask for real-time display.

```
def handle_hashtag(rdd):
    print("hashtag", str(datetime.now()))
    try:
        url = base_url + '/update_tweet_tags'
        temp = rdd.sortBy(lambda x: x[1], False).take(10)
        temp = {k: v for k, v in temp}
        ans = {"labels": list(temp.keys()), "counts": list(temp.values())}
        print(ans)
        response = requests.post(url, json=ans)
    except Exception as e:
        print(e)
```

```
windowLength = 60 * 60
slideInterval = 10


lines = lines.flatMap(lambda x: x.lower().split("t_end")).filter(lambda x: len(x)>0)

sentiments = lines.transform(transform_rdd_sentiment).reduceByKeyAndWindow(lambda a,b:a+b, None, windowLength, slideInt

tags = lines.transform(transform_rdd_hashtag).reduceByKeyAndWindow(lambda a,b:a+b, None, windowLength, slideInterval)

sentiments.foreachRDD(handle_sentiment)

tags.foreachRDD(handle_hashtag)
```

```
ssc.start()
```

Figure 23: pyspark_stream.ipynb

Flask

A web frontend has been built using Flask for displaying real-time statistics. A dashboard page with the total analysed tweet data and the topic mentioned most will be displayed in a real-time basis. The port exposed is 5000, such that any data exchange between pyspark to Flask, and Flask to the user will through this port.

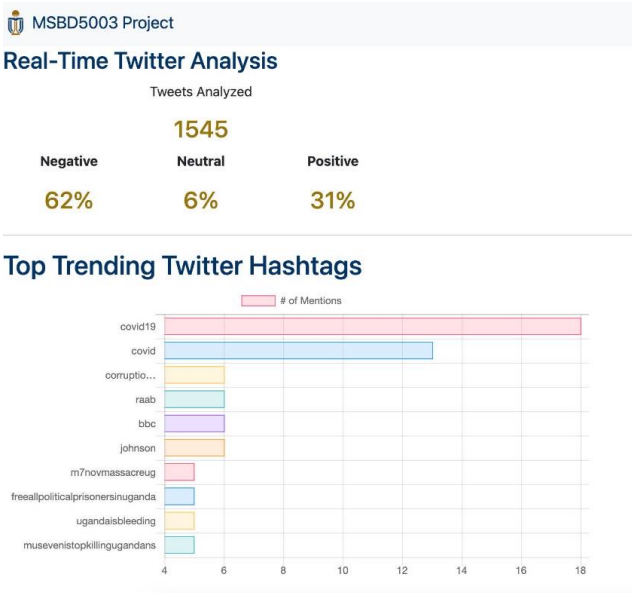


Figure 24: Real-Time Twitter Analysis Webpage

There are mainly two POST request handlers, and one GET request handler.
GET request

- Directory: "/" - It will return the dashboard html file to the user.

POST request

- Directory: "/update_tweet_counters" - It will receive the POST json data of total tweet counts analysed.
- Directory: "/update_tweet_tags" - It will receive the POST json data of the titles and its counts.

For each spark stream batch, it will POST the data after received from Twitter, and perform analysis. Flask will automatically update the data, and display on the dashboard.

Summary

Discussion

In this project, we have exploited some big-data-related technologies, here is a small summary about their advantages compared with conventional methods.

	Big Data	**Conventional**
Data Storage	- Tb to Pb - Distributed File System, e.g. DBFS. The capacity depends on all nodes. - More reliable. N copies on nodes and automatic recovery if data loss.	- Gb to Tb - Local File System. The capacity depends on the size of the local hard disk. - Have to perform hard disk data recovery if the hard disk is broken.
Data Processing and Analysis	- Parallelism, e.g. Spark SQL, Spark NLP, Spark MLlib - Example 1: Spark NLP v.s. NLTK. We have tried to use NLTK to pre-process the tweets text but it is so slow. We eventually switch to Spark NLP since it can process tasks such as removal of stopwords and lemmatization parallelly based on Spark while NLTK is only optimized in sequential programing. - Example 2: Spark MLlib. While it is hard to use machine learning libraries like sk-learn on a large dataset (not efficient and out of memory). Spark MLlib performs machine learning algorithms parallelly and can handle much bigger datasets.	- Sequential
System	- Cloud Computing - Horizontal Scaling - Dynamic provision. For example, we can tune the cluster size freely in Databricks - PaaS. Databricks is user-friendly to	- Local Computing - Vertical Scaling

	those who are new to Spark and cloud clusters. It is so easy to set up the workplace for our project.	

Table 6: Big Data and Conventional Methods Comparison

Future Work

Due to limitations of time and computational power, we've only focused on COVID tweets in the US for this project. There are many other interesting areas where we can further develop in the future. Our ideas for future work include but are not limited to the following,

- Building a graph between the retweets and learning how sentiments affect one another. Virus is not the only thing that can transmit from one person to another, emotions can as well. It would be interesting to use spark graphframe to further break down the tweets trend and see how different types of emotions such as anger, happiness, sadness, etc. are retweeted and learn whether these tweets with high retweet numbers share certain common traits.
- Analyzing more regions, countries, ethnic and social groups. Due to limited time, we've only studied the public response as a whole in the US. However, different cultural and social backgrounds usually cultivate different kinds of online behaviors. We are curious to find out how tweet sentiment varies in other parts of the world, for example, whether sentiments are more neutral in a more conservative society such as Japan.
- Allowing others to access the past data. Aside from utilizing spark streaming to show real time twitter statistics, we can perhaps include past data into our webpage to allow for a more holistic display of the whole COVID timeline.

Conclusion

COVID 19 is still happening worldwide. Knowledge gained from the trend and effect of the pandemic are still important in helping the society. In this project, both real time and past twitter data have been retrieved for analysis. Emotions, word counts, both topic modeling and sentiment analysis are derived from the large twitter data, and mainly negative events are mostly found until now.

Also, we would like to thank **Professor Yi Ke**, and all the Teaching Assistants of MSBD 5003 for the support of this project, helping us in using Big Data Technology on the project discovery and data handling. Last but not least, we hope that the pandemic will pass, and people get back to normal life soon.

Reference List

[1]Gupta, R., Vishwanath, A., & Yang, Y. (2020, July 18). *Global reactions to covid-19 on Twitter: A labelled dataset with latent topic, sentiment and emotion attributes*. openICPSR. Retrieved October 13, 2021, from https://www.openicpsr.org/openicpsr/project/120321/version/V10/view?path=%2Fopenicpsr%2F120321%2Ffcr%3Aversions%2FV10%2FTwitter-COVID-dataset---Sep2021%2FCOVID_Twitter_database_paper.pdf&type=file

[2]https://github.com/owid/covid-19-data

[3]*Spark Streaming Programming Guide*. Spark Streaming - Spark 3.1.2 Documentation. (n.d.). Retrieved October 13, 2021, from https://spark.apache.org/docs/latest/streaming-programming-guide.html