Twitter Data Analysis on COVID-19

by

Group 8: Sakshi Agrawal, Sahana Bhat, Vidyashree Lakshminarasimhan

IMT 563: Advanced Database Management Systems

Information School
UNIVERSITY of WASHINGTON

OVERVIEW

On 11th March 2020, the World Health Organization announced COVID19 outbreak as a pandemic. Starting from China, this virus has infected and killed thousands of people from Italy, Spain, USA, Iran and other European countries as well. While this pandemic has continued to affect the lives of millions, a number of countries have resorted to complete lockdown. During this lockdown, people have taken social networks to express their feelings and find a way to calm themselves down.

The Corona Virus endangers our physical health indeed, but alongside, social distancing also poses a threat to our emotional stability. Thus, it is crucial to understand public sentiments under COVID-19.

This analysis has been done to analyse how people are dealing with the situation. The tweets have been collected, pre-processed, and then used for text mining and sentiment analysis.

OBJECTIVES

1. Find out the topics most discussed about in the tweets
2. Analyze the sentiment of the tweets
3. Analyze the most tagged User IDs
4. Future scope: Analyze misinformation associated with the tweets about the pandemic

METHOD AND DESIGN

Tools and Technologies:

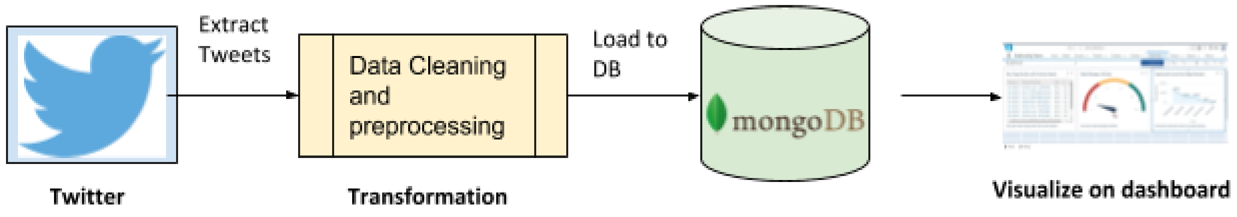
Since we are dealing with unstructured data, we are planning on using MongoDB as our NoSQL Database.

In order to insert data into the MongoDB database, we are using MongoShell and Python drivers. This will enable us to utilize Python's data manipulation packages to do our data analysis and basic visualization.

We are using Atlas as the Cloud service provider to host our data on the cloud.

For creating our Dashboards, we used the MongoDB Charts feature which is a powerful visualization tool offered by MongoDB for visualising the collections and documents created.

Conceptual Diagram:

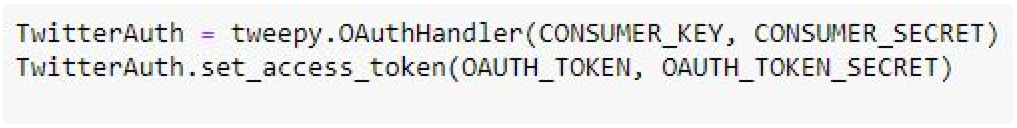


ETL PIPELINE

Twitter data extraction using Twitter API

To extract data from Twitter, we use the API's provided by the company. The Twitter Search API allows us to grab only a certain number of historical tweets. We used the Streaming API, that allows us to stream data in real-time for analysis and grabs a much larger dataset than the Search API. The extracted tweets are filtered based on the hashtags related to COVID.

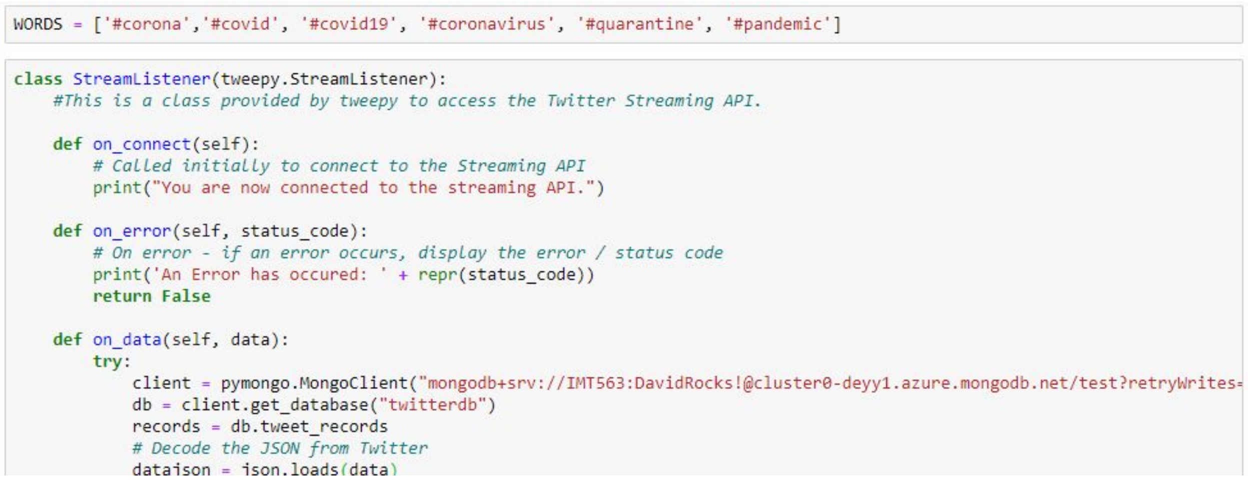
A Twitter App was created, to get access to our personal consumer key, consumer secret, access token and access token secret. These special API keys help with user Authentication when extracting data from the Streaming API.



```
TwitterAuth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
TwitterAuth.set_access_token(OAUTH_TOKEN, OAUTH_TOKEN_SECRET)
```

Steam Listener

To extract Tweets from the Streaming API, the Tweepy StreamListener class is used. The on_data function connects to the defined mongoDB () and stores the tweets as JSON data. Any errors and exceptions are captured by on_error function.(Brown, 2017)



```
WORDS = ['#corona','#covid', '#covid19', '#coronavirus', '#quarantine', '#pandemic']
```

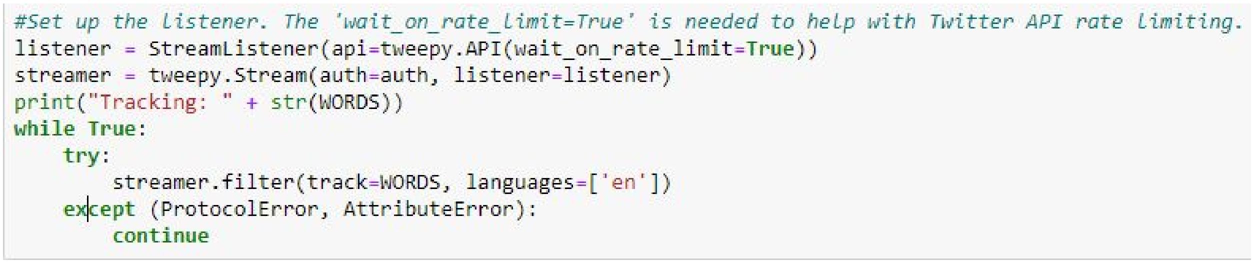
```
class StreamListener(tweepy.StreamListener):
    #This is a class provided by tweepy to access the Twitter Streaming API.

    def on_connect(self):
        # Called initially to connect to the Streaming API
        print("You are now connected to the streaming API.")

    def on_error(self, status_code):
        # On error - if an error occurs, display the error / status code
        print('An Error has occured: ' + repr(status_code))
        return False

    def on_data(self, data):
        try:
            client = pymongo.MongoClient("mongodb+srv://IMT563:DavidRocks!@cluster0-deyy1.azure.mongodb.net/test?retryWrites=
            db = client.get_database("twitterdb")
            records = db.tweet_records
            # Decode the JSON from Twitter
            datajson = json.loads(data)
```

Below code shows setting up the Twitter Stream Listener. The tweets are filtered based on hashtags related to covid.



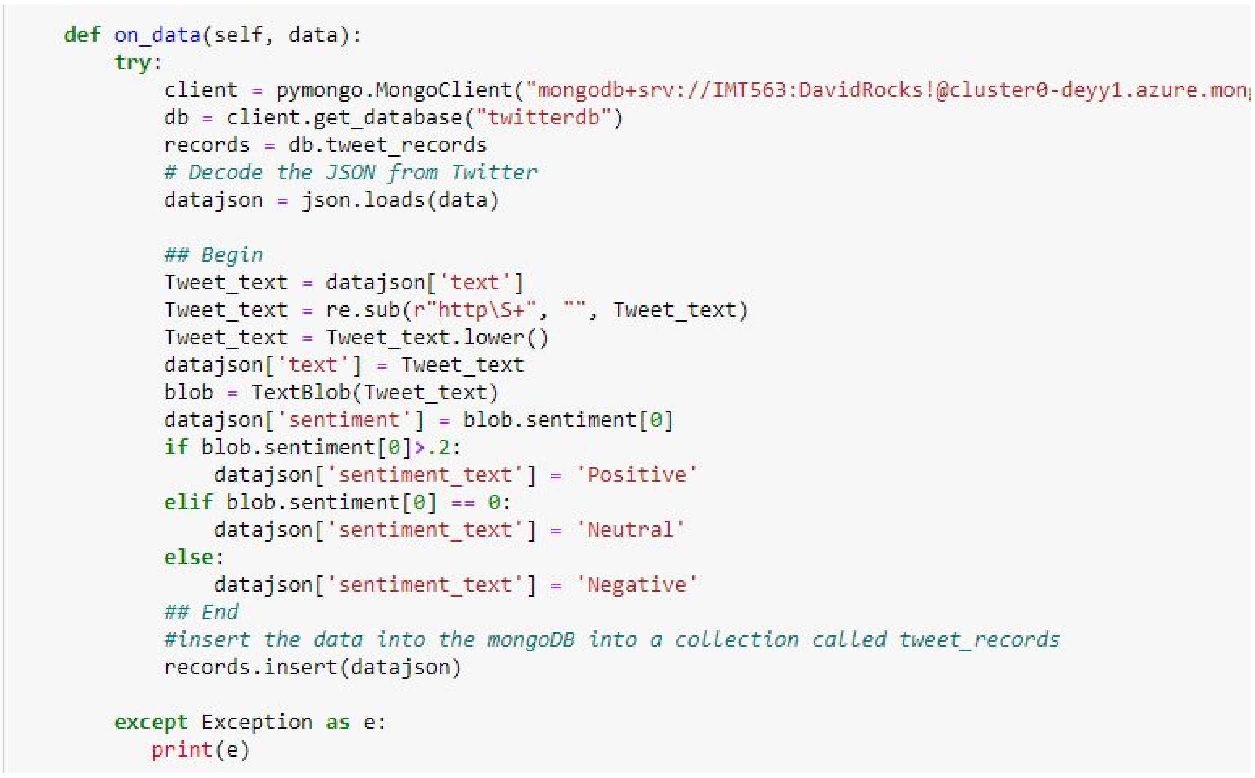
```
#Set up the listener. The 'wait_on_rate_limit=True' is needed to help with Twitter API rate limiting.
listener = StreamListener(api=tweepy.API(wait_on_rate_limit=True))
streamer = tweepy.Stream(auth=auth, listener=listener)
print("Tracking: " + str(WORDS))
while True:
    try:
        streamer.filter(track=WORDS, languages=['en'])
    except (ProtocolError, AttributeError):
        continue
```

Data Preparation and Loading into MongoDB

Mongo Client is created using pyMongo. The connection string is provided which contains the MongoDB repository name and password. The Mongo Client helps us to connect with the Database and the Collection we have created. Each Tweet data obtained on the stream is inserted into the Collection in Json format.

When the data arrives in the Stream, the Tweet data is read into Json format to perform cleaning. The following data preprocessing steps were implemented:

- The Tweet text was extracted, special characters were removed and the text was converted to lowercase.
- **Sentiment Analysis:** The Tweet text was used to obtain a sentiment score using TextBlob library. The sentiment_text variable is created and assigned values: Positive, Negative or Neutral based on the sentiment score obtained.



```
def on_data(self, data):
    try:
        client = pymongo.MongoClient("mongodb+srv://IMT563:DavidRocks!@cluster0-deyy1.azure.mon
        db = client.get_database("twitterdb")
        records = db.tweet_records
        # Decode the JSON from Twitter
        datajson = json.loads(data)

        ## Begin
        Tweet_text = datajson['text']
        Tweet_text = re.sub(r"http\S+", "", Tweet_text)
        Tweet_text = Tweet_text.lower()
        datajson['text'] = Tweet_text
        blob = TextBlob(Tweet_text)
        datajson['sentiment'] = blob.sentiment[0]
        if blob.sentiment[0]>.2:
            datajson['sentiment_text'] = 'Positive'
        elif blob.sentiment[0] == 0:
            datajson['sentiment_text'] = 'Neutral'
        else:
            datajson['sentiment_text'] = 'Negative'
        ## End
        #insert the data into the mongoDB into a collection called tweet_records
        records.insert(datajson)

    except Exception as e:
       print(e)
```

DATABASE IMPLEMENTATION

We chose MongoDB as a Database of choice because we were dealing with unstructured data in the form of JSON files. With a NoSQL database, we got the ability to store the JSON files as is into the database.

The structure of a typical JSON will look like:

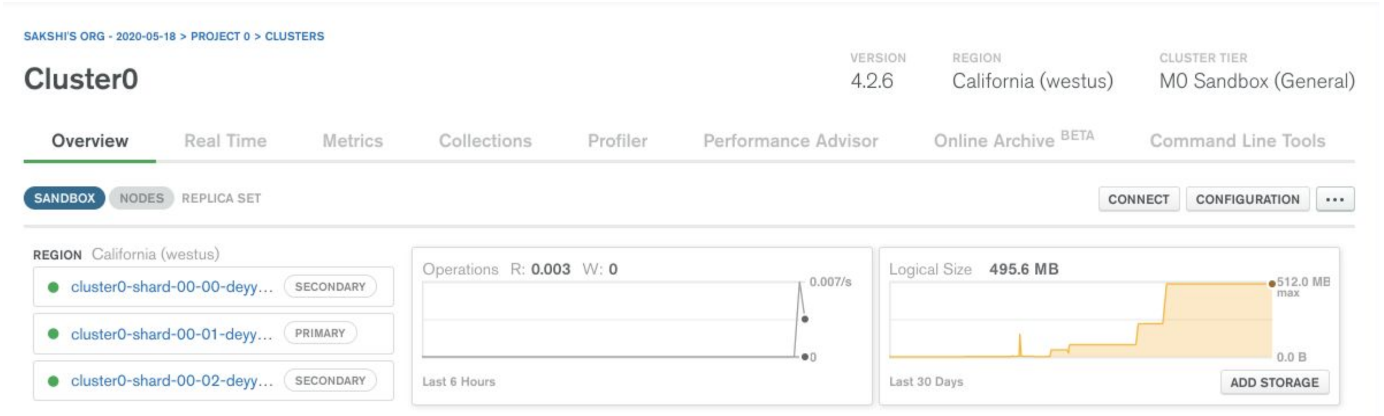
{

TweetID,

```
        Created_at,
        Text,
        User_id,
        User_name,
        Hashtags,
        URL,
        Tagged_User,
        Geo,
        Coordinates,
        Reply_count,
        Retweet_count
}
```

We created a single Collection called **‘Twitterdb’** to store our data.

To host the Database on cloud, we used Atlas on MongoDB.
We created a M0 Sandbox cluster which allows to store 500 MB of data free:



SAKSHI'S ORG - 2020-05-18 > PROJECT 0 > CLUSTERS

Cluster0

VERSION 4.2.6

REGION California (westus)

CLUSTER TIER M0 Sandbox (General)

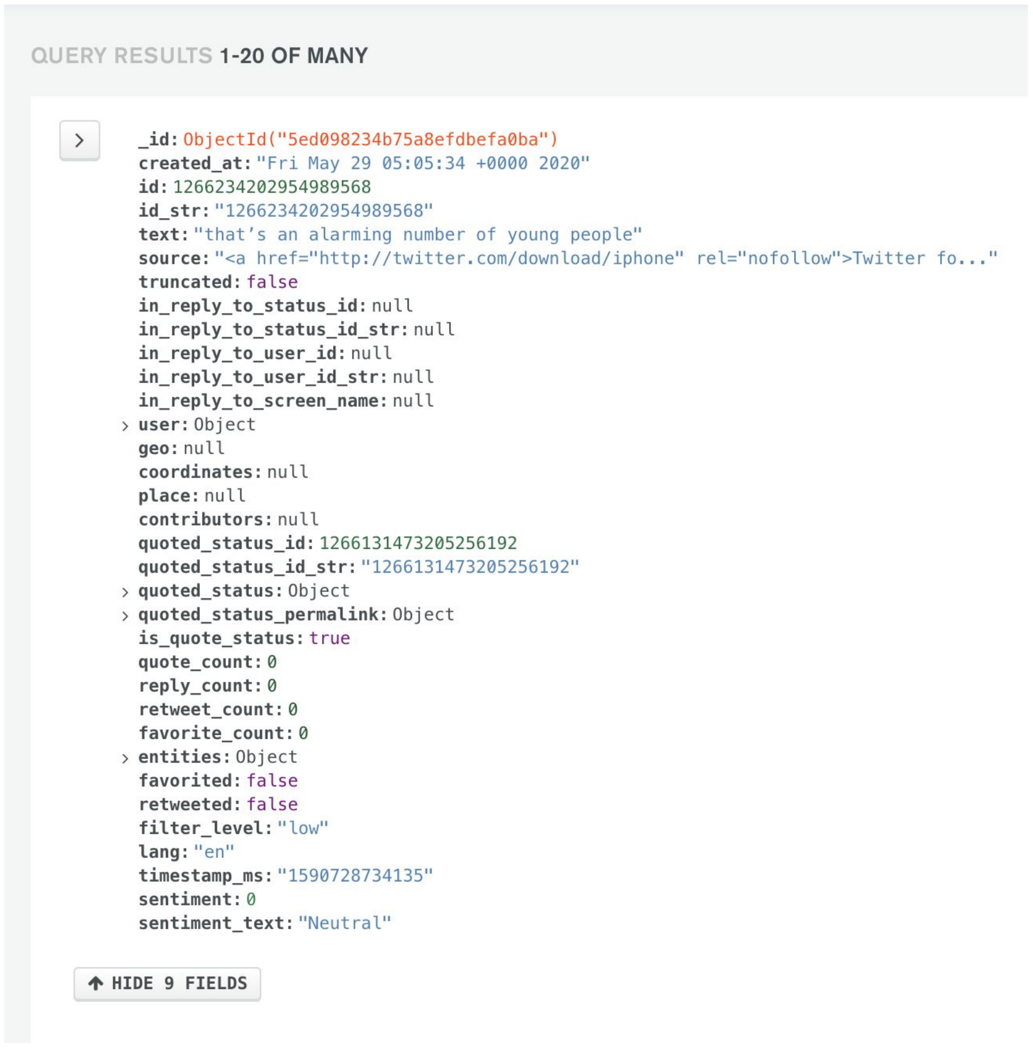
REGION California (westus)

cluster0-shard-00-00-deyy… SECONDARY
cluster0-shard-00-01-deyy… PRIMARY
cluster0-shard-00-02-deyy… SECONDARY

Operations R: 0.003 W: 0

Logical Size 495.6 MB

And this is how our JSON files look like:


QUERY RESULTS 1-20 OF MANY

```
_id: ObjectId("5ed098234b75a8efdbefa0ba")
created_at: "Fri May 29 05:05:34 +0000 2020"
id: 1266234202954989568
id_str: "1266234202954989568"
text: "that’s an alarming number of young people"
source: "<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter fo..."
truncated: false
in_reply_to_status_id: null
in_reply_to_status_id_str: null
in_reply_to_user_id: null
in_reply_to_user_id_str: null
in_reply_to_screen_name: null
> user: Object
geo: null
coordinates: null
place: null
contributors: null
quoted_status_id: 1266131473205256192
quoted_status_id_str: "1266131473205256192"
> quoted_status: Object
> quoted_status_permalink: Object
is_quote_status: true
quote_count: 0
reply_count: 0
retweet_count: 0
favorite_count: 0
> entities: Object
favorited: false
retweeted: false
filter_level: "low"
lang: "en"
timestamp_ms: "1590728734135"
sentiment: 0
sentiment_text: "Neutral"
```

VISUALIZING DATA USING MONGODB CHARTS

Why MongoDB Charts instead of Tableau for visualization?

The unique feature in the case of MongoDB Charts is that it visualizes the data without having to resort to flattening JSON data, to transform to a relational structure. MongoDB Charts can query individual fields without having to merge them all into a single row, or explode the document and represent nested fields in separate relational tables. To simplify and shorten the distance between data and insights for all data: unstructured, structured and multi-structured, we decided to accomplish this by exploring on the MongoDB Charts which does not restrict us to mapping to a relational structure.(Loughead, n.d.)
Tableau is an excellent analytics tool for structured relational data. The MongoDB BI Connector used to connect MongoDB data to Tableau, is moving data out of MongoDB into structured tables so that Tableau works. MongoDB is a powerful database solution for modern data.

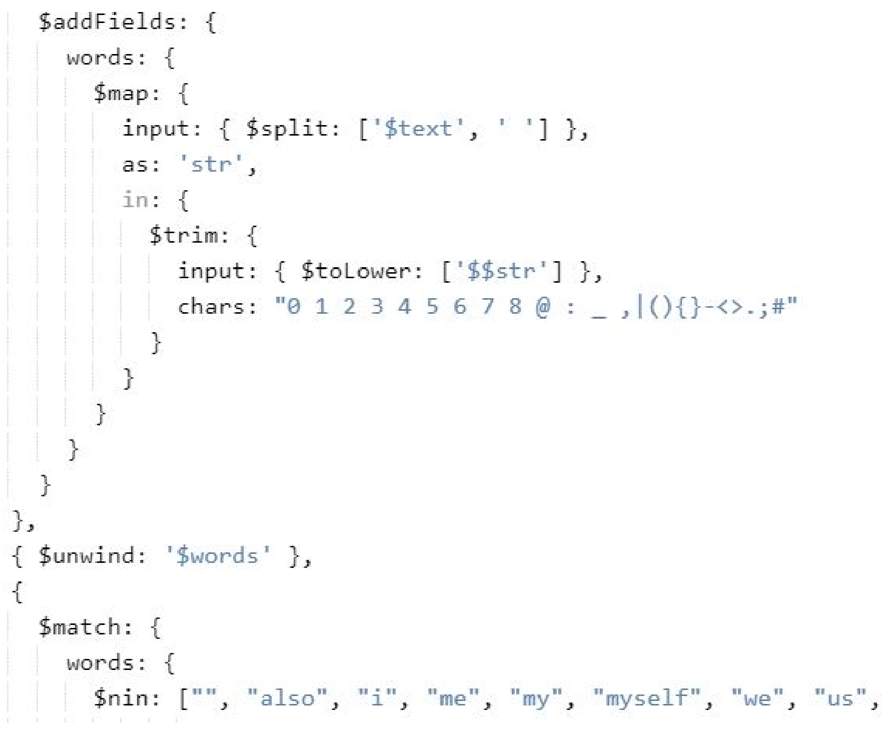
Moving data out of MongoDB into a structured relational database for analytics seems to defeat the purpose of the investment.

MongoDB Aggregation Pipeline to build Charts

The MongoDB Aggregation Framework provides developers with express functional pipelines that support data preparation, transformations, and analysis. This is performed in stages using specific actions for data grouping, matching, sorting, or shaping. Data flowing through the various stages and its processing is known as the Aggregation Pipeline. This is very similar to the data flow through a Unix shell command line pipeline. Data gets input from the previous stage, work is performed and the stage's output serves as input to the next processing stage until the pipeline ends.(Walters, 2019)

The MongoDB shell script used to perform text analysis on the tweet are:

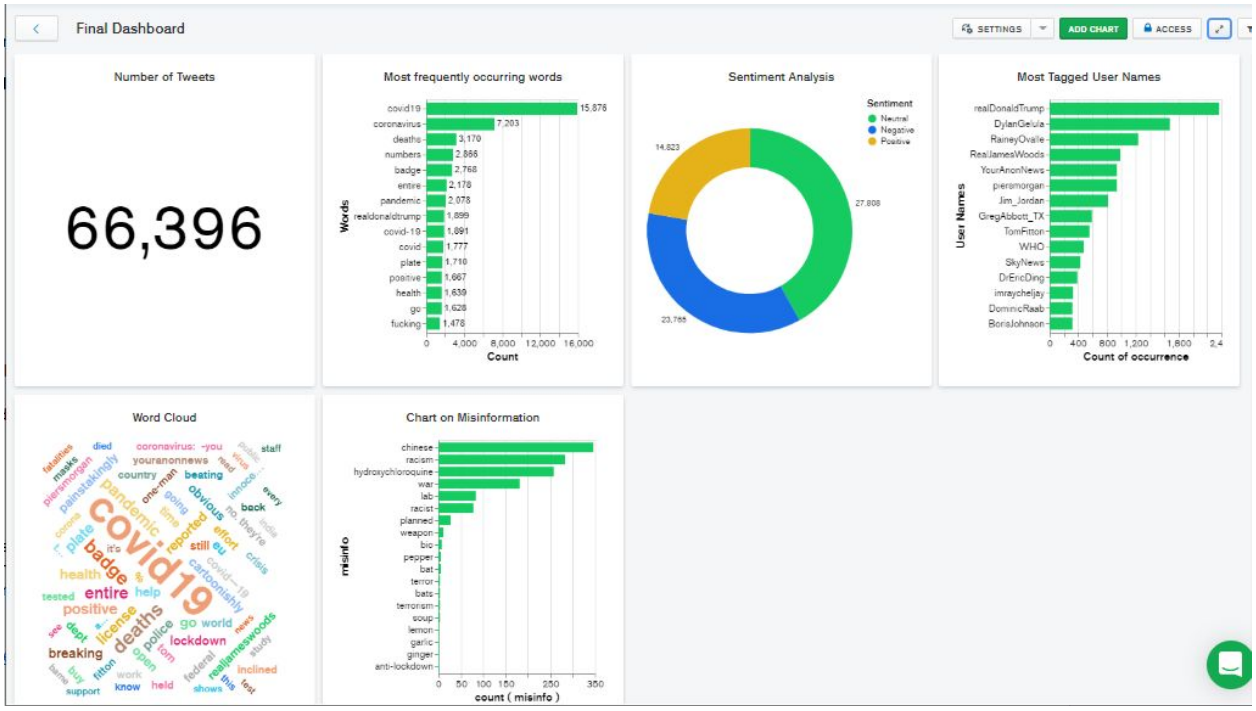
- **$addfields:** It is used to add a new field in the document.
- **$trim:** It removes white spaces and null.
- **$unwind:** It can deconstruct an array field from the input documents to output a document for each element. Each output document is the input document with the value of the array field replaced by the element.
- **$nin:** It means 'Not In'.It just returns the documents without the mentioned words. We used it to remove unnecessary words.



```
  $addFields: {
    words: {
      $map: {
        input: { $split: ['$text', ' '] },
        as: 'str',
        in: {
          $trim: {
            input: { $toLower: ['$$str'] },
            chars: "0 1 2 3 4 5 6 7 8 @ : _ ,|(){}-<>.;#"
          }
        }
      }
    }
  }
},
{ $unwind: '$words' },
{
  $match: {
    words: {
      $nin: ["", "also", "i", "me", "my", "myself", "we", "us",
```

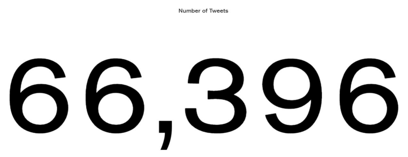
Visualizations

We created a dashboard with a collection of visualizations based on the analyses performed. The below screen represents the dashboard with the collection of charts.

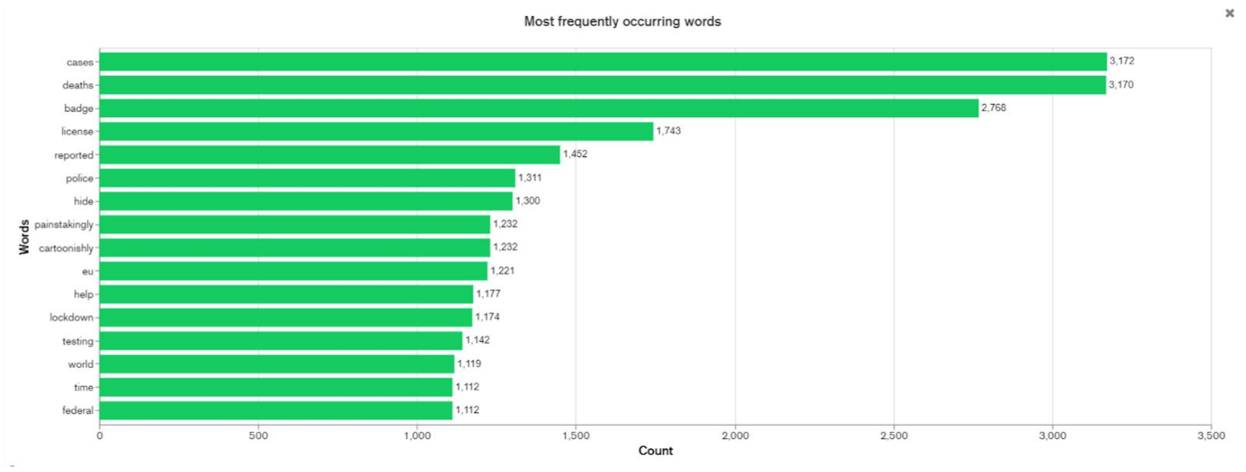


The visualizations created are as follows:

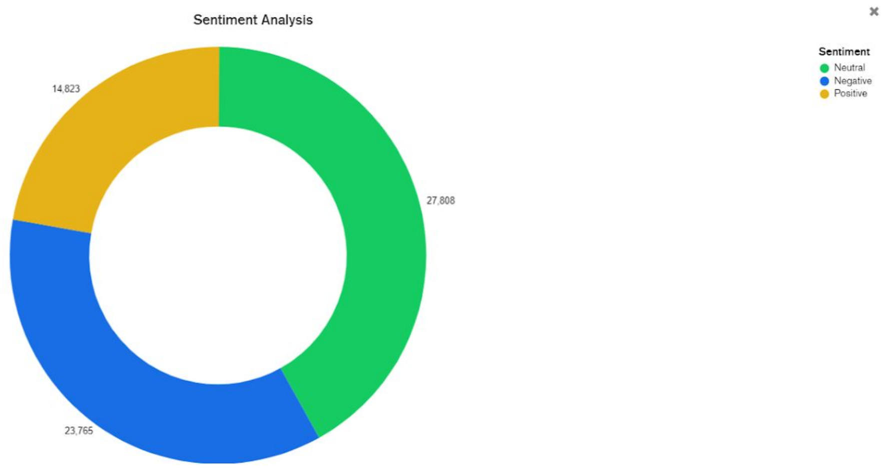
1. **Number of Tweets:** Represents the count of the tweets streamed using the Twitter API and stored in MongoDB for further analysis.



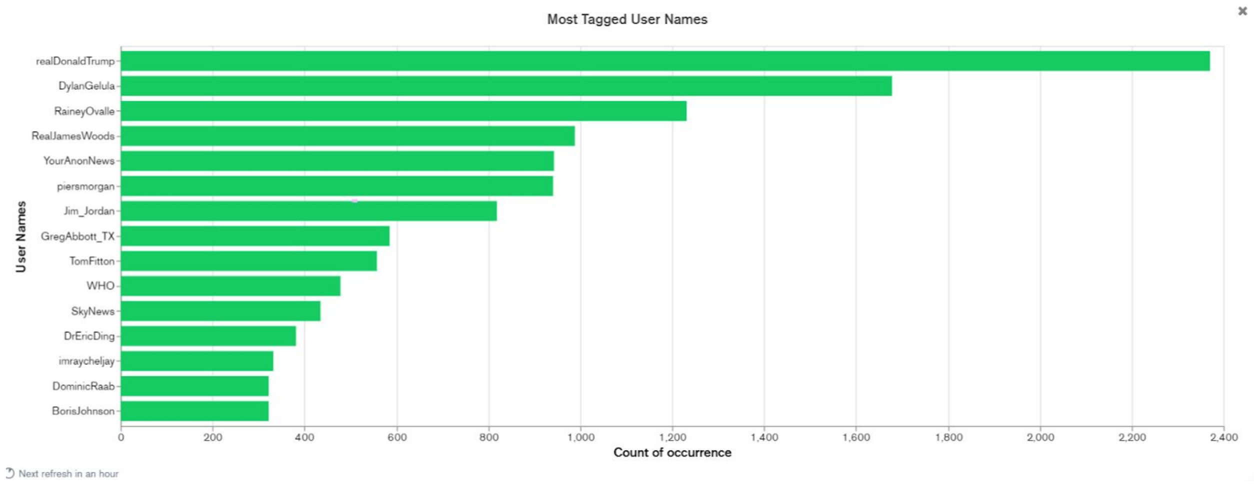
2. **Most frequently occurring words:** A bar chart on the count of most frequently tweeted words during the course of pandemic.



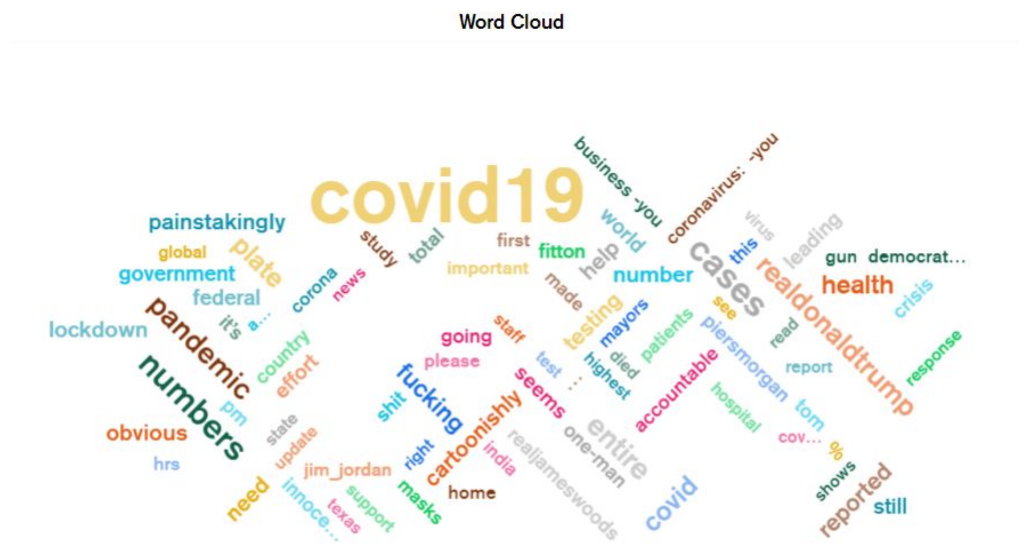
3. **Sentiment Analysis:** Sentiment analysis was carried out using textblob. It is a simple API which returns polarity and subjectivity. Polarity which can be mapped as positive,negative or neutral sentiments. It was used to create a donut chart.



4. **Most Tagged User Names:** An analysis on the most tagged username in Twitter over the past few months.

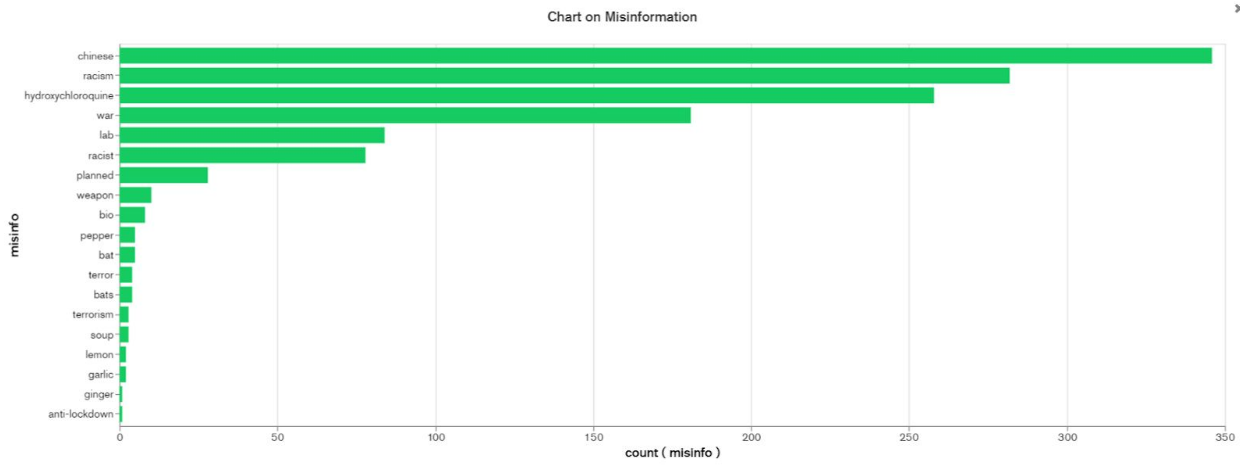


5. **Word Cloud:** A basic text analysis using MongoDB shell scripts to identify the commonly used words that spread misinformation.



A word cloud to depict the most searched keywords during the time of the pandemic(COVID19).

6. **Chart on Misinformation:**



LIMITATIONS AND CHALLENGES

Some of the limitations and challenges we faced during the course of this project have been highlighted below.

- With the free tier of Atlas, we got only 500 MB of space.This in turn limited our ability to perform comprehensive text analysis on a larger amount of tweet data .
- Cleaning the tweets before performing the analysis was a major challenge we faced as the tweets included data such as emojis or emoticons which are hard to filter out,

especially when we were creating word clouds.

- We wanted to see how sentiment changed over time through the pandemic, but with limited tweets that wasn't possible. This was another limitation of the free tier of the MongoDB Atlas feature. Studying the change in sentiments over a period of time will definitely be a part of our future scope.
- We started with an intent to analyze misinformation as well, but couldn't build a model as yet for that. There is scope for future research in this as we aim to use machine learning models and natural language processing toolkits to come up with more precise and suggestive words regarding misinformation. We would also like to think about ways to reduce the spread of misinformation, if possible.
- The Tweets we could extract did not have location enabled for most of them, hence we could not do any Geospatial analysis. Most of the tweets we scraped did not have location information as the user had disabled it. We wanted to analyze the tweets country wise which was not possible because of this limitation.

REFERENCES

Walters, R. (2019, September 19). Retrieved from https://www.mongodb.com: *https://www.mongodb.com/blog/post/time-series-data-and-mongodb-part-3--querying-analyzing-and-presenting-timeseries-data*

Loughead, K. (n.d.). Tableau and MongoDB Analytics: Why It's a Bad Marriage. Retrieved from https://blog.knowi.com: https://blog.knowi.com/2018/03/tableau-and-mongodb-analytics.html

Brown, E. D. (2017, January 19). Collecting / Storing Tweets with Python and MongoDB. Retrieved from https://pythondata.com/collecting-storing-tweets-with-python-and-mongodb/

APPENDIX

The link to view our python implementation code:

https://github.com/SahanaBhat263/IMT563/blob/master/TwitterStreamingFinal.ipynb

The link to view our final dashboard:

https://charts.mongodb.com/charts-project-0-ggoer/public/dashboards/55a34c4f-1401-4165-af1b-0e304f889258