IBM HC 2020
- Covid-19 sentiment Analysis

Team - We are Groot

Aditya Kumar
Kesava Karri
Nipun Haldar
P. R. Rahul Hebbar

Problem Statement

Sentiment Analysis of COVID-19 Tweets – Visualization Dashboard

The sentiment analysis of Indians after the extension of lockdown announcements to be analyzed with the relevant #tags on twitter and build a predictive analytics model to understand the behavior of people if the lockdown is further extended.Also develop a dashboard with visualization of people reaction to the govt announcements on lockdown extension

Pain-Point

- Sentiment Analysis of COVID-19 Tweets.
- Visualization Dashboard

Our Approach

We can split the entire process needed for creating the intuitive Dashboard into smaller sections as follows.

- Data Fetching
- Data Pre-processing
- Creating Backend
- Designing Frontend
- Deployment

Continuation

Data Fetching:

Here we will be running a python script that will be listening continuously to the twitter stream and fetch us the tweet along with various other parameters relating to the tweet like user info, location, date and time created and more

Data Pre-processing:

In this section we will be using the data got from the above tweet listener and performing three actions on the tweet data them being data cleaning, sentiment analysis and database entering. The database will be entered with all the necessary parameters we have.

Continuation

Creating Backend:

We have the database with all the necessary parameters in it. We will be now reading the database here and extracting all the parameters to make cumulative calculations and then returning the refined parameters got to the frontend using socket connection.

Designing Frontend:

The Frontend designing is done using react.js and the cross platform ability supported by electron.js. We will be establishing socket connection with our backend and receiving continuous packets of data needed for visualization.

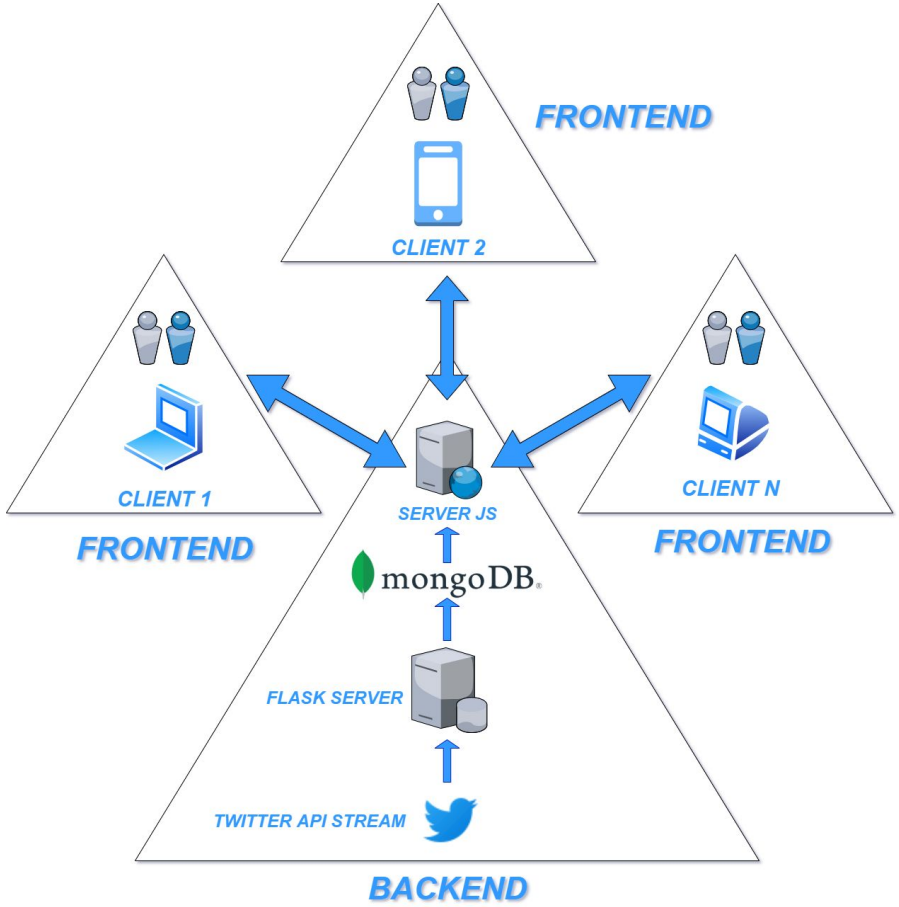
Deployment:

We will be deploying the Frontend and Backend on IBM cloud.

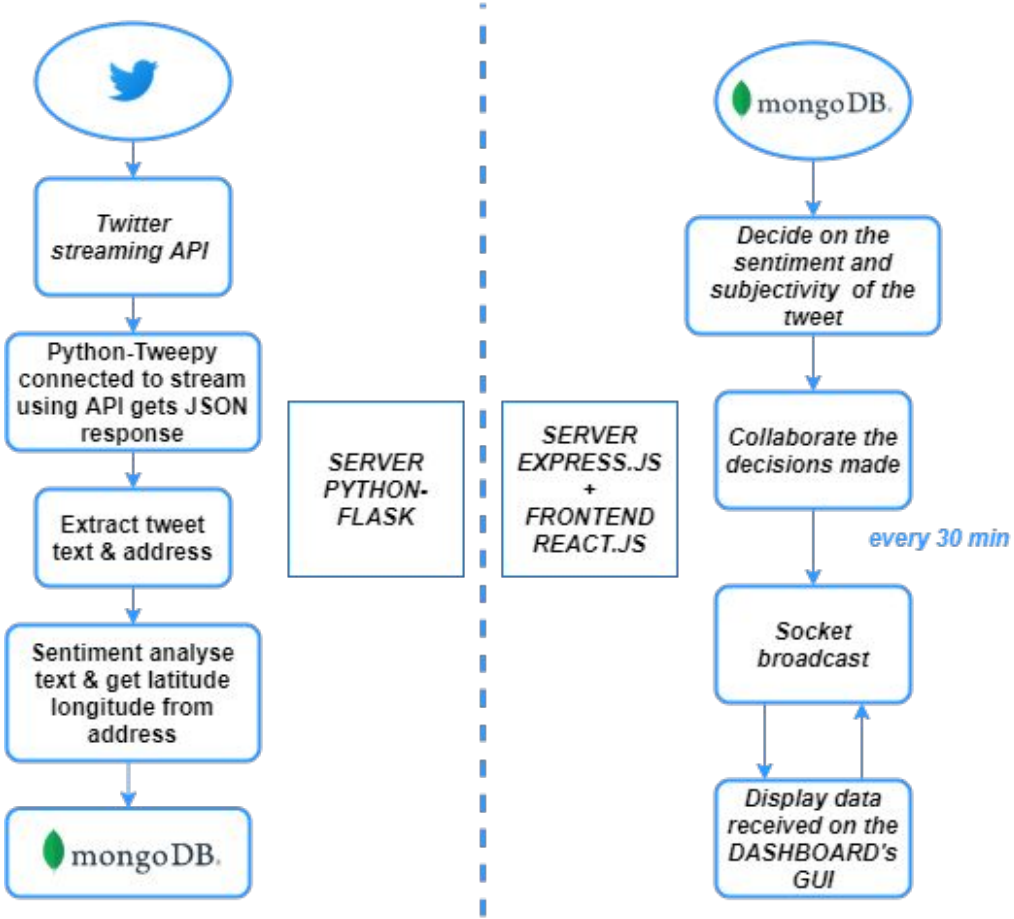
Significance

- Multi Platform compatibility.
- Open-Source throughout.
- Easy to install installers.

Architecture



Process Flow



Flowchart explanation

Our Flowchart has two sections, the Backend Server implemented using Flask and the Backend + Frontend implemented using JavaScript. Let us look at the flow in detail

Python-Flask Server

- We will be using the twitter's streaming API made for developers and provide filter parameters such that we get the resent COVID 19 related tweets made by Indians.
- Python Tweepy library is used to connect to api twitter stream by supplying four keys - API key, API key secret, Service token and Service Token Secret to OAuth protocol. After OAuth get authorized we get access to live stream updates based on passed parameters. Every new tweet is given to use with lot of parameters like created at string, timestamp, unique tweet-id, user information, address etc, as a json string.

Continuation

- The json response we receive has too many parameters that we don't need. So we will be using only the main and necessary parameters which include created at string, unique tweet id, tweet text and tweet location address parameter.
- Once we have all the necessary parameters in our hand, we will perform operations on the tweet text and tweet location. We use Unidecode to remove emojis and unwanted character and then use the cleaned text with Textblob to extract the polarity (sentiment score) and subjectivity (emotional content score). And for forward geo-coding (getting latitude and longitude from address), we use the Nominatim OpenStreetMaps http request method to get the most probable latitude and longitude for the given address. Here it is important to note that we pass country code IN (ISO 3166-1 alpha-2 format country code for INDIA) to get only latitude and longitude of places within INDIA and for other address get null.
- Once we process the parameters we send these refined results to MongoDB Atlas database as json documents containing tweets created at string, tweet id, polarity score, subjectivity score, latitude and longitude.

Continuation

EXPRESS.JS Server and Frontend

- Our server to fetch data from MongoDB and consolidate it is written in JavaScript. We are creating a backend server using express.js to establish a two way socket.io websocket connection between the backed js with the frontend dashboard.
- We connect to our MongoDB database every 30 minutes and fetch all the new update using MongoDB find queries and a checkpoint tweet id.
- Once we receive the new updates in the stream from the database, we run few decision instructions on polarity and subjectivity score to segregate the updates into positive, neutral or negative tweets and objective or subjective tweet.

Continuation

- After segregating the decisions, we collaborate the data in such a way that the object when sent to our frontend will get easily processed by graph library. We maintain the collaborated data in a checkpoint.json file for newly connected users and recovery from failures. The socket.io by default broadcasts the new updates to all our connected clients every 30 minutes.
- The frontend is written using React.js with GUI's components and styling using Material.ui, interactive graphs using Nivo library and location plot support using Mapbox's API for javascript.

Top Features

- Web framework for visualization using React.js.
- Python-Flask, MongoDB Atlas and Express.js Integration.
- Stream Twitter API for live updates using Python-Tweepy.
- Sentiment and Emotion analysis of tweet text using Python Textblob.
- Location Plot of Tweets.
- Interactive and Intuitive Dashboard.
- End-to-end application deployed on IBM Cloud using Cloud Foundry.

Technologies Used

- Python
  - Flask
  - Tweepy
  - Textblob
  - Unidecode
  - Requests
  - Urllib
  - Pymongo
  - Multiprocessing
- IBM Services
  - Cloud Foundry
- API
  - Twitter
  - OpenStreetMaps
  - Mapbox
- Javascript
  - React.js
  - Electron.js
  - Socket.io
  - Mongodb
  - Dotenv
  - Nivo
  - Material-ui
  - Mapbox-gl

THANK YOU