**TEAM NAME: BOT-1**

**Team Leader: Tejaswi Pallapothu**

**Application ID: SPS\_CH\_APL\_20200002303**

**PROJECT NAME: Sentiment Analysis of Covid-19 Tweets**

**GITHUB LINK:** https://github.com/tejaswi-29/SBSPS-Challenge-1125-Sentiment-Aanalysis-of-Covid-19-tweets---Visualisation-Dashboard.git

TABLE OF CONTENTS:

1) INTRODUCTION

a) OVERVIEW

b) PURPOSE

2) LITERATURE SURVEY

a) EXISTING PROBLEMS

b) PROPOSED SOLUTION

3) THEORITICAL ANALYSIS

a) BLOCK DIAGRAM

b) HARDWARE/SOFTWARE DESIGNING

4) EXPERIMENTAL INVESTIGATIONS

5) FLOWCHART

6) RESULT

7) ADVANTAGES AND DISADVANTAGES

8) APPLICATIONS

9) CONCLUSION

10) FUTURE SCOPE

11) BIBLIOGRAPHY

APPENDIX AND REFERENCES

LINK TO GITHUB REPOSITORY

1. **INTRODUCTION**
   1. **OVERVIEW**

**Sentiment Analysis is the process of determining whether a piece of writing is positive, negative or neutral.** A sentiment analysis system for text analysis combines natural language processing (NLP) and machine learning techniques to assign weighted sentiment scores to the entities, topics, themes and categories within a sentence or phrase.

The sentiment analysis of Indians after the extension of lockdown announcements will be analysed with the relevant #tags on twitter and a dashboard showing the sentiments of the tweets, with visualizations of people’s reaction to the government’s decisions of lockdown will be built to understand the behaviour of people regarding COVID-19.

* 1. **PURPOSE**

Results the solution will bring:

1. Get to know people’s sentiment towards the pandemic.

2. Understand the sentiments of people on the government's decision to extend the lockdown.

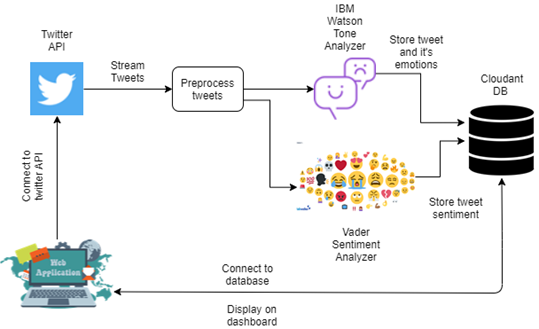
This analysis has been done to analyse how the citizens of India are dealing with the situation.

1. **LITERATURE SURVEY**
   1. **EXISTING PROBLEM**

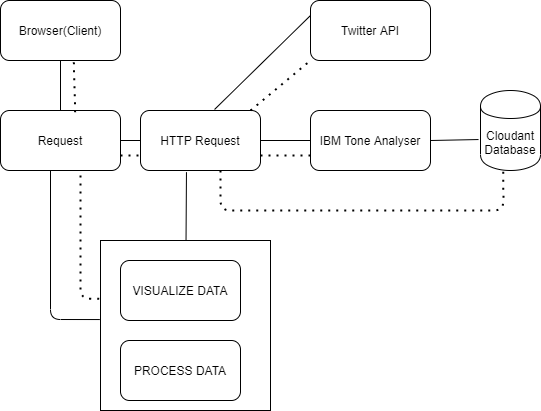
The main problem existing in the current techniques of most of the sentiment analysis solutions is that they only give the output as positive, negative, neutral or just the score i.e. if the score is greater than zero, it will be considered positive, zero then its neutral else it is negative. Also, most of the algorithms cannot deal with complex sentences that require more than sentiment words and simple analysing.

* 1. **PROPOSED SOLUTION**
* The solution for analysing the sentiments of the tweets in a better way is to use IBM Watson’s tone analyser because it tries to understand the tone and emotion of the sentence. There are currently 7 different tones: Joy, Fear, Anger, Confident, Sadness, Tentative, Analytical. Along with Watson Tone Analyser, Vader sentiment analyser which is a rule-based sentiment analyser will be used to define a tweet as negative, positive or neutral.
* Tweets will be collected using Twitter API and they processed and passed into Tone Analyzer and Vader, the tweet along with its tone and sentiment and other properties is stored in the Cloudant database.
* But Tweepy, the library to connect to the Twitter API, has a limitation: The tweets that are more than a week old cannot be collected. So, GetoldTweets3 is a library that allows to scrape data from twitter that are older than 1 week without requiring an API key. So, the tweets from each of the lockdowns can be collected using this library.
* For each lockdown, 1000 tweets will be stored in my local system because the cloudant does not have enough storage.
* For lockdown 1: stream tweets until 15th April 2020
* For lockdown 2: stream tweets until 3rd May 2020
* For lockdown 3: stream tweets until 17th May 2020
* For lockdown 4: stream tweets until 31st May 2020
* For lockdown 5: stream tweets until 9th June 2020
* These tweets will then be sent to Tone Analyser to identify the tweet emotion and the count of each emotion and the aggregate sum of score of each emotion score will be stored in Cloudant database.
* For the live sentiment detection, the tweets along with their sentiments and emotion tones will be stored in Cloudant.
* The dash application will display the graphs of the sentiment dynamically and calculate the percentage of the positive, negative and neutral tweets along with the most frequently used words and the counts of each emotion and the latest tweets are also displayed.

1. **THEORITICAL ANALYSIS**
   1. **BLOCK DIAGRAM**



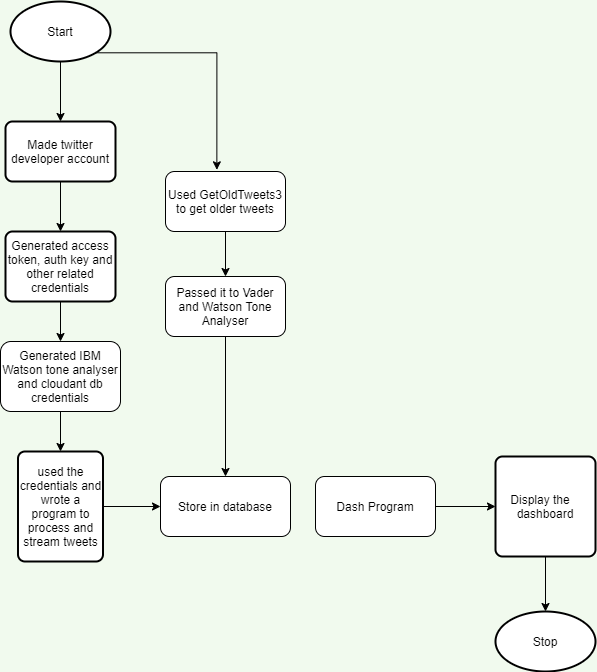
* 1. **HARDWARE/SOFTWARE DESIGNING**



1. **EXPERIMENTAL INVESTIGATIONS**

* In the process of making this dashboard, I have learnt to use IBM cloud services, how to use its database and Tone Analyser and learnt Plotly Dash web framework. Using IBM’s automated services for Machine Learning and sentiment analysis, it has become a lot easier than ever before to detect sentiment. I also learnt to use Zoho writer and Kanban board for updating this project report.
* After building the dashboard UI, I have noticed that a lot of people are actually happy with the lockdown because of the fact that they can better utilise their time to improve their skills so the emotion which was the highest is **“Joy”.**
* The second most common emotion I noticed was “**Sadness**”. This pandemic has caused a lot number of troubles and deaths in families so far and yet to cause many till a cure is actually found. So, a lot of people’s tweets were about grievances.
* The third most common emotion is **“Fear”** because given the current scenario, a lot number of people from different industries are facing layoffs and the most affected people are the daily wage workers. Also, many people’s tweets showed the fear of losing a loved one.
* Usually without certain major situations like covid-19 happening the numbers of positive and negative tweets are relatively close and stay low compared with neutral tweets.
* And neutral tweets are about 20% every time, this shows a significant change in how people express their feelings.
* About 45-50% of the tweets are positive which suggests that people are fighting this in a diligent way. They are looking forward to making best use of their time and hoping for a speedy recovery which also indicates that they are agreeing on the decision taken by the government although it is very difficult to follow because its nearly not possible to completely stay at home but it is essential.
* About 25-30% of the tweets are negative which includes sadness and anger showing the problems that has been caused.
* Also there were a lot of tweets which showed critisizing China calling Covid-19 as Chinese virus this also contributed to the negative sentiment.
* The analysis also showed that joy kept on decreasing as each lockdown passed by. This shows people are unhappier than before.

1. **FLOWCHART**

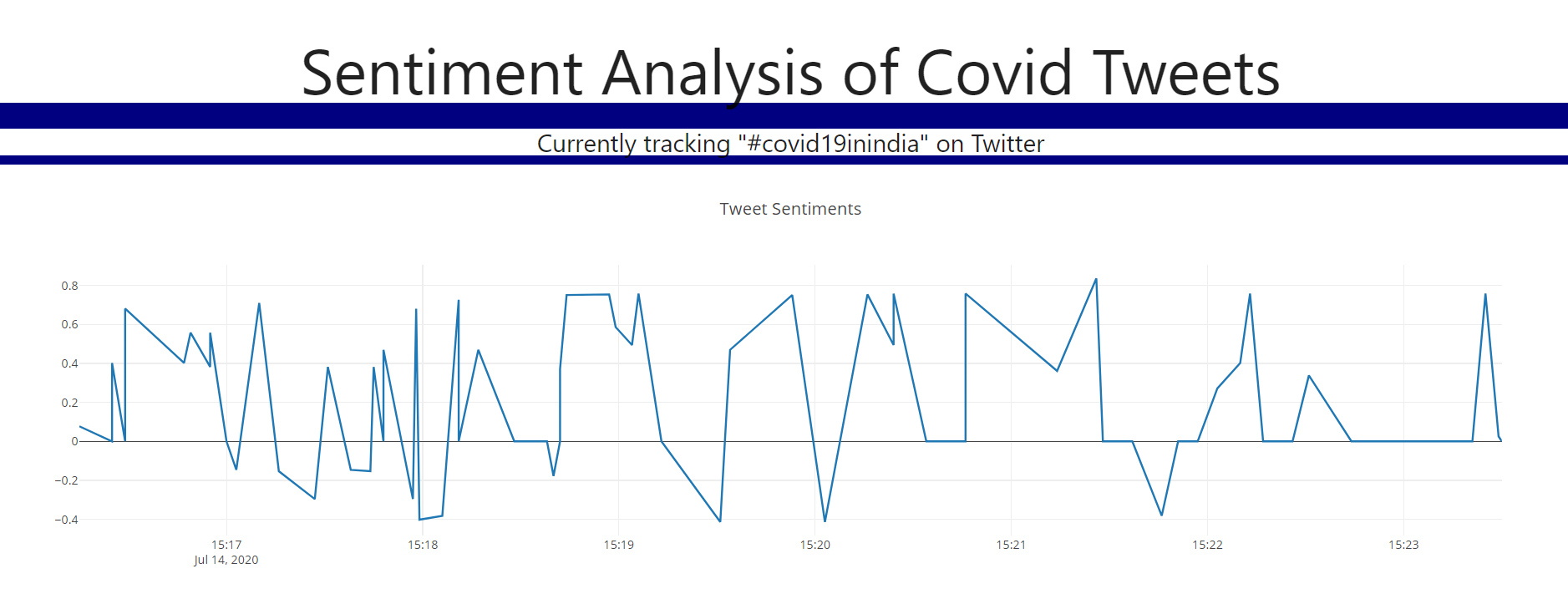


1. **RESULT**

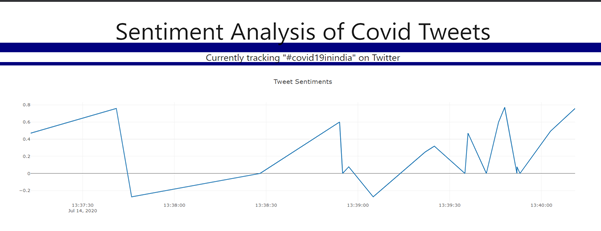
**UI:**

**Tweets live sentiment at different times.**

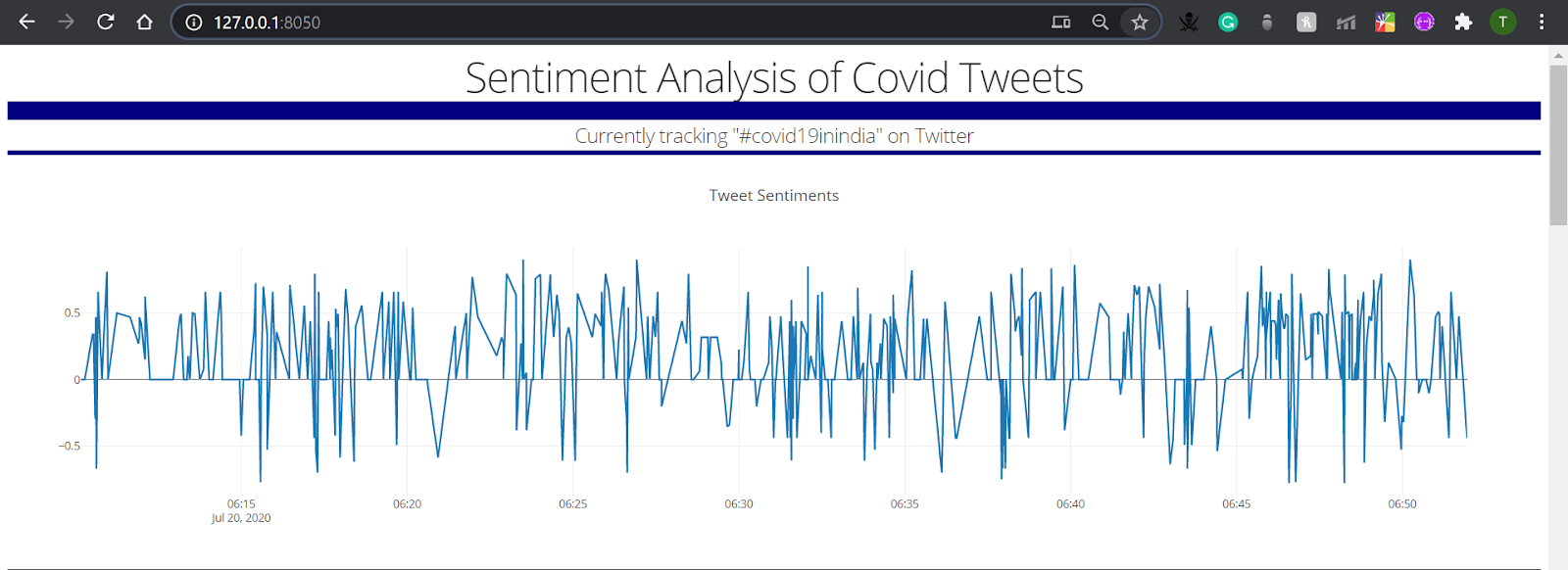
14th July at 3 PM

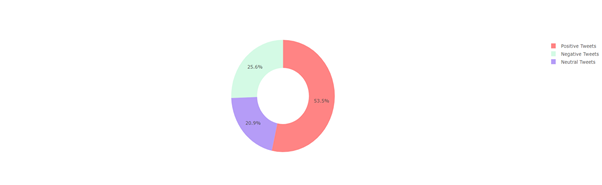
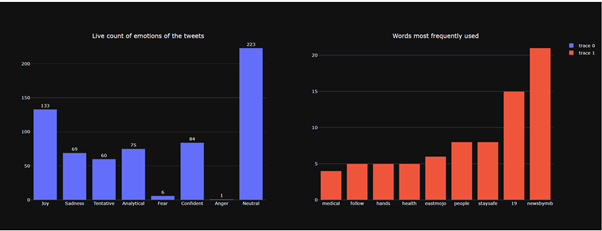


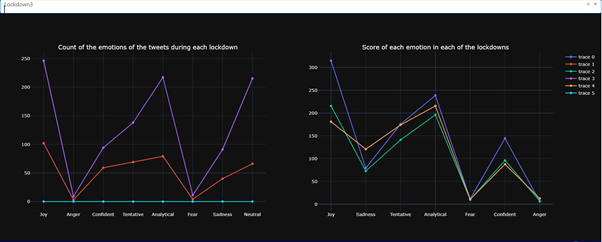
14th July at 1 PM



20th July at 6:15 AM





Details of each lockdown

Words Most Frequently used in each of the 5 lockdowns.





1. **ADVANTAGES AND DISADVANTAGES**

**Advantages:**

* The Vader sentiment analyser gives only if the overall score of a sentence whether it’s positive, negative or neutral. But using IBM Watson Tone analyser, 7 new tones (Joy, Anger, Confident, Tentative, Sadness, Analytical, Fear) have been added to better analyse the tweets.
* The benefits of analysing the tweets to understand public sentiment helps us to understand what actually people think about this pandemic and what are the reactions towards the same.
* This analysis can help identify what the crisis is in a particular region by simply changing the track words in the twitter streaming program for e.g. #mumbaicovid or #delhifightscorona.
* We can get all the tweets related to this hashtag and the dashboard will display what the emotions of the public are and what the most commonly used words are in a set.
* Also, for example: a state government decides to set up a quarantine centre but is not sure which region to keep, it can take help of public sentiment region wise along with of course where the cases are very high and then finalise the place or city.

**Disadvantages:**

* Some tweets like: “The number of cases is rising day by day” is negative in this context, is characterized positive because the word “rise” is generally a positive term and the analyser does not understand the context of such sentences.
* Some of the tones like “Sarcasm” are not detected by both Vader and Tone Analyser.
* IBM cloud is a paid service
* The database requires a lot of space for storing the tweets and its associated values. Since the Cloudant DB has only 2gb of free space, I could'nt store all the tweets.

1. **APPLICATIONS**

* This dashboard can be used to find the sentiment of the people in each of the lockdowns.
* This analysis can help identify what the crisis is in a particular region by simply changing the track words in the twitter streaming program for e.g. #mumbaicovid or #delhifightscorona
* Also, for example: a state government decides to set up a quarantine centre but is not sure which region to keep, it can take help of public sentiment region wise along with of course where the cases are very high and then finalise the place or city.

1. **CONCLUSION**

I enjoyed doing every bit of this project right from the start, setting up timelines and learning about the IBM services. This web app detects the sentiment of people in India and can be used sentiment worldwide. I thank, IBM and Smartbridge for its free services to help students in their projects.

1. **FUTURE SCOPE**

This project has been able to understand the overall live sentiment of people on Covid-19 and also the sentiment of people in each of the 5-country wide lockdowns

Tweets are collected using Twitter API and they processed and passed into Tone Analyzer and Vader, the tweet along with its tone and sentiment and other properties is stored in the Cloudant database. The dash application displays the graphs of the sentiment dynamically and calculates the percentage of the positive, negative and neutral tweets along with the most frequently used words and the counts of each emotion and the latest tweets are also displayed.

This application can be further used to display the live count of number of Covid-19 cases in each state of India and also throughout the world. Also, sentiment analysis can be done for the world as well to know what is the crisis in every part. Some countries change their news when displaying it to the public i.e. the actual situation that is happening in the region is different than that is being shown in the news but, Twitter is a place where thoughts are expressed and the real situation can be actually found out using the sentiment analysis. So, this project can be used to show the actual situation.

11. **BIBLIOGRAPHY AND REFERENCES**:

* IBM Cloudant documentation: <https://python-cloudant.readthedocs.io/en/stable/getting_started.html#identity-and-access-management-iam>
* IBM Watson Tone Analyser: <https://cloud.ibm.com/apidocs/tone-analyzer>
* Creating twitter developer account: <https://docs.inboundnow.com/guide/create-twitter-application/>
* Streaming live tweets: <https://youtu.be/1gQ6uG5Ujiw>
* Getting Old Tweets: <https://pypi.org/project/GetOldTweets3/>
* Sentiment Analysis using Vader: <https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664>
* Dash documentation: <https://dash.plotly.com/>
* Sentiment analysis using Dash: <https://pythonprogramming.net/live-graph-twitter-sentiment-analysis-gui-dash-python/>
* Deploying dash app on IBM Cloud: <https://dash.plotly.com/deployment>

<https://moderndata.plot.ly/hosting-dash-apps-on-ibm-cloud/>

* **LINK TO GITHUB REPOSITORY:** https://github.com/tejaswi-29/SBSPS-Challenge-1125-Sentiment-Aanalysis-of-Covid-19-tweets---Visualisation-Dashboard.git

**APPENDIX**

**a) SOURCE CODE:**

**This is the main code for getting the historical tweets:**

import GetOldTweets3 as got

import pandas as pd

tweetCriteria = got.manager.TweetCriteria().setQuerySearch('#lockdownextended').setNear("Mumbai").setLang('en').setUntil("2020-04-30").setEmoji('unicode').setMaxTweets(1000)

tweets = got.manager.TweetManager.getTweets(tweetCriteria)

**This is the main code for streaming live tweets and storing them in cloud:**

def on\_data(self, data):

#This is the meat of the script...it connects to your mongoDB and stores the tweet

try:

datajson = json.loads(data)

# print(datajson)

tweet = unidecode(datajson['text'])

tweet = tweet.lower()

tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)

tweet = re.sub(r'#([^\s]+)', r'\1', tweet)

print("done preprocessing")

print(tweet)

tone\_analysis = tone\_analyzer.tone({'text': tweet},content\_type='application/json').get\_result()

tone=json.dumps(tone\_analysis, indent=2)

jsonParse = json.loads(tone)

vs = analyzer.polarity\_scores(tweet)

sentiment = vs['compound']

emotions\_score = {}

if "document\_tone" in jsonParse.keys():

if jsonParse["document\_tone"]["tones"]!=[]:

length = len(jsonParse["document\_tone"]["tones"])

k = jsonParse["document\_tone"]['tones']

for i in range(length):

n = k[i]['tone\_name']

my\_document[n] += 1 #updating the score count of the emotion

my\_document.save()

print("document updated!")

if n in emotions\_score:

emotions\_score[n] = emotions\_score[n]+k[0]['score']

print("done adding the score")

else:

emotions\_score.update({n:k[0]['score']})

print("adding a new emotion")

else:

emotions\_score={"tone":"None"}

my\_document['Neutral']+=1

my\_document.save()

else:

emotions\_score={"tone":"None"}

my\_document['Neutral']+=1

my\_document.save()

global df

global tweetdetails

my\_document['Tweet\_Count'] +=1

tweetdetails.update({my\_document['Tweet\_Count']:[datajson['created\_at'],datajson['text'],sentiment]})

# df = df.append({'Created\_at':datajson['created\_at'],'tweet':datajson['text'],'Emotions':emotions\_score},ignore\_index = True)

my\_document['Sentiment\_score\_avg']=(my\_document['Sentiment\_score\_avg']+sentiment)/my\_document['Tweet\_Count']

my\_document['Tweets'] = tweetdetails

my\_document.save()

print("Updated document!")

**This is the source code for the main dash application:**

import pandas as pd

import dash

import dash\_table

import dash\_core\_components as dcc

import plotly.express as px

import dash\_html\_components as html

import plotly.graph\_objects as go

from plotly.subplots import make\_subplots

from dash.dependencies import Output,Input

from dash\_table.Format import Format

import dash\_table.FormatTemplate as FormatTemplate

import datetime

import time

import re

import nltk

nltk.download('punkt')

nltk.download('stopwords')

from nltk.probability import FreqDist

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

# from database import df

from cloudant.client import Cloudant

from cloudant.error import CloudantException

from cloudant.result import Result, ResultByKey

client = Cloudant("username", "password", url="url")

client.connect()

print("connected to db!")

my\_database = client['tweets2']

my\_document = my\_database['Livefeed']

lockdown = my\_database['Lockdowns']

#initalising app

external\_stylesheets = ['https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css']

app = dash.Dash(\_\_name\_\_,external\_stylesheets=external\_stylesheets)

app.title = 'Tweet Monitor'

df1 = pd.DataFrame()

df2 = pd.DataFrame()

for i in my\_document['Tweets'].values():

df1 = df1.append({'Created\_at':i[0],'tweet':i[1],'Sentiment':i[2]},ignore\_index = True)

df1['Created\_at'] = pd.to\_datetime(df1['Created\_at'])

#finding most frequently used words

content = ' '.join(df1["tweet"])

content = re.sub(r"http\S+", "", content)

content = content.replace('RT ', ' ').replace('&amp;', 'and')

content = re.sub('[^A-Za-z0-9]+', ' ', content)

content = content.lower()

content = content.replace("july", ' ')

content = content.replace("covid",' ')

tokenized\_word = word\_tokenize(content)

stop\_words=set(stopwords.words("english"))

filtered\_sent=[]

for w in tokenized\_word:

if w not in stop\_words:

filtered\_sent.append(w)

fdist = FreqDist(filtered\_sent)

fd = pd.DataFrame(fdist.most\_common(10),columns = ["Word","Frequency"]).drop([0]).reindex()

#counting negatives and positives and neutrals:

slist = df1['Sentiment'].tolist()

positive=0

negative=0

neutral=0

for i in slist:

if i<0:

negative+=1

elif i>0:

positive+=1

else:

neutral+=1

polaritycounts = [positive,negative,neutral]

polaritynames = ["Positive Tweets","Negative Tweets","Neutral Tweets"]

#getting the data table

def get\_data\_table():

table = pd.DataFrame()

table['Tweet']=df1['tweet']

table['Sentiment']=df1['Sentiment']

data\_table = dash\_table.DataTable(

id = 'datatable-data',

columns = [{'name':c,'id':c} for c in table.columns],

data = table.to\_dict('records'),

page\_current = 0,

page\_size = 20,

# page\_actions = 'custom',

# style\_table = {'overflowY':'scroll'},

style\_as\_list\_view=True,

# style\_as\_list\_view=True,

style\_header={'backgroundColor': 'rgb(30, 30, 30)',

'width':30,

'textAlign': 'center'},

style\_cell={

'backgroundColor': 'rgb(50, 50, 50)',

'color': 'white'

},

fixed\_rows={'headers': True, 'data': 0},

style\_table={'minHeight': '600px',

'height': '600px',

'maxHeight': '600px',

'overflowX': 'auto',

'width':'auto'

},

sort\_action="native",

sort\_mode="multi",

column\_selectable="single"

)

return data\_table

#creating graphs:

y=[my\_document['Joy'],my\_document['Sadness'],my\_document['Tentative'],my\_document['Analytical'],my\_document['Fear'],

my\_document['Confident'],my\_document['Anger'],my\_document['Neutral']]

graphs = make\_subplots(rows=1,cols=2,

subplot\_titles=("Count of the emotions of the tweets during each lockdown","Score of each emotion in each of the lockdowns"))

graphs1 = make\_subplots(rows=1,cols=2,subplot\_titles=("Live count of emotions of the tweets","Words most frequently used"))

graphs1.update\_layout(height=600,template="plotly\_dark")

trace1 = go.Bar(

x=['Joy','Sadness','Tentative','Analytical','Fear','Confident','Anger','Neutral'],

y=y,

text = y,

textposition = 'outside'

)

graphs1.append\_trace(trace1,1,1)

trace2 = go.Bar(

y=fd["Frequency"].loc[::-1],

x=fd["Word"].loc[::-1],

orientation='v'

)

graphs1.append\_trace(trace2,1,2)

l1 =go.Scatter(

x=['Joy','Sadness','Tentative','Analytical','Fear','Confident','Anger'],

y=[314.877,79.450,175.787,238.881,11.576,144.614,5.759]

)

l1count = go.Scatter(

y = [v for v in lockdown['Lockdown1'].values()],

x = [v for v in lockdown['Lockdown1']]

)

l2 = go.Scatter(

x=['Joy','Sadness','Tentative','Analytical','Fear','Confident','Anger'],

y=[215.541,72.755,141.158,196.285,9.922,96.045,6.924],

)

l2count = go.Scatter(

y = [v for v in lockdown['Lockdown2'].values()],

x = [v for v in lockdown['Lockdown2']]

)

l3 = go.Scatter(

x=['Joy','Sadness','Tentative','Analytical','Fear','Confident','Anger'],

y=[180.970,120.723,174.059,215.677,10.581,87.614,12.564]

)

l3count = go.Scatter(

y = [v for v in lockdown['Lockdown3'].values()],

x = [v for v in lockdown['Lockdown3']]

)

l4 = go.Scatter(

x = ['Joy','Sadness','Tentative','Analytical','Fear','Confident','Anger'],

y = [153.572,85.073,154.209,213.389,9.925,106.957,5.418]

)

l4count = go.Scatter(

y = [v for v in lockdown['Lockdown4'].values()],

x = [v for v in lockdown['Lockdown4']]

)

l5 = go.Scatter(

x = ['Joy','Sadness','Tentative','Analytical','Fear','Confident','Anger'],

y = [219.869,79.320,132.351,192.129,9.346,94.883,11.068]

)

l5count = go.Scatter(

y = [v for v in lockdown['Lockdown5'].values()],

x = [v for v in lockdown['Lockdown5']]

)

app.layout = html.Div(

[

html.H1('Sentiment Analysis of Covid Tweets',

style={

'textAlign':'center',

'background': 'white'

}

),

html.H5('Currently tracking "#covid19inindia" on Twitter',

style={

'textAlign':'center',

'background':'white'

}),

html.Div([

dcc.Graph(

id ='graph-1',

animate=True,

figure={

'data':[

go.Scatter(

x=df1['Created\_at'],

y=df1['Sentiment']

)

],

'layout':{

'title':'Tweet Sentiments',

'hovermode':"closest",

'xaxis\_title':"Created at",

'yaxis\_title':"Sentiment score"

}

}

)

]),

dcc.Graph(

figure=graphs1,id='graph2n3'

),

dcc.Graph(

id = 'donut',

figure={

'data':[

go.Pie(labels=polaritynames,

values=polaritycounts,

hole=0.50,name='View Metrics!',

marker\_colors = ['rgba(255, 50, 50, 0.6)','rgba(184, 247, 212, 0.6)','rgba(131, 90, 241, 0.6)']),

]

}

),

html.Div(children = [

dcc.Dropdown(

id= 'dropdown',

options=[

{'label': 'Lockdown1', 'value': 'lockdown1'},

{'label': 'Lockdown2', 'value': 'lockdown2'},

{'label': 'Lockdown3', 'value': 'lockdown3'},

{'label': 'Lockdown4', 'value': 'lockdown4'},

{'label': 'Lockdown5', 'value': 'lockdown5'},

]

),

dcc.Graph(

figure=graphs,id='graph4'

)]),

html.Div(children=[html.H1(children="Live Tweets", # html for table

style={

'textAlign': 'center',

"background": "white",

'font-size':60

}),

get\_data\_table()])

], style = {

'background':'#000080'

}

)

# if value=='None':

@app.callback(

dash.dependencies.Output('graph4', 'figure'),

[dash.dependencies.Input('dropdown', 'value')])

def display\_graphs(value):

if value=='lockdown1':

graphs.append\_trace(l1,1,2)

graphs.append\_trace(l1count,1,1)

if value=='lockdown2':

graphs.append\_trace(l2,1,2)

graphs.append\_trace(l2count,1,1)

if value == 'lockdown3':

graphs.append\_trace(l3,1,2)

graphs.append\_trace(l3count,1,1)

if value == 'lockdown4':

graphs.append\_trace(l4,1,2)

graphs.append\_trace(l4count,1,1)

if value == 'lockdown5':

graphs.append\_trace(l5,1,2)

graphs.append\_trace(l5count,1,1)

graphs['layout'].update(height=600,template="plotly\_dark")

return graphs

#Running the app

if \_\_name\_\_ == '\_\_main\_\_':

app.run\_server(debug=True)