IBM Hackathon 2020

Final Project Report

Sentiment Analysis of Covid-19 Tweets

Rahul R

Vishnu Anil

INDEX

1 Introduction

Twitter is a popular microblogging service where users create status messages (called "tweets"). These tweets sometimes express opinions about different topics. Due to the outbreak of CoronaVirus, twitter has become a popular platform to express individuals' views on the current scenario.

The purpose of this project is to build an algorithm that can accurately classify Twitter messages as positive, negative or neutral, with respect to a query term. Our hypothesis is that we can obtain high accuracy on classifying sentiment in Twitter messages using machine learning techniques.

1.1 Purpose

Assessing the reaction of a country's population to various topics is a mammoth task even for well organized government bodies. In an era where data governs the world, each individual has the freedom to express their opinions through microblogging services like Twitter. The project is aimed at analysing these reactions and put to use by Government bodies to make better governing decisions and for other sections of the community to understand how people are reacting to an event/product/decision.

1.2 Overview

The application is developed to analyse the sentiment in the tweets posted during the various lockdown phases enforced by the government due to the outbreak of Covid-19 and to classify the tweets based on the sentiment as Positive, Negative or Neutral. The analysis will provide an insight into people's opinion about the various topics related to Covid-19 and can be used to predict how people would react if the scenario is to continue and hence formulate decisions accordingly. The application will be supported by a dashboard for ease of understanding through visualization.

The application will include:

- Visual comparison of the various sentiments expressed by the people through various phases of lockdown.
- Visual comparison on the most trending topics and hashtags through the various phases of lockdown.
- A prediction model to understand how the sentiment will be in the future if the same scenario is to continue.

2 Literature Survey

2.1 Existing problem

Gathering and analysing the reaction of a country's population to various events occuring in the country requires a lot of time and space without proper visual aids. There is great difficulty to categorize the slangs and emojis of a lot of data with the existing techniques . Going through billions of reactions will require a large amount of time and hence is impossible to be done by humans. The data has to be collected, sorted, analysed and classified based on the sentiment. The result after analysis will be impossible to understand without proper visualization techniques.

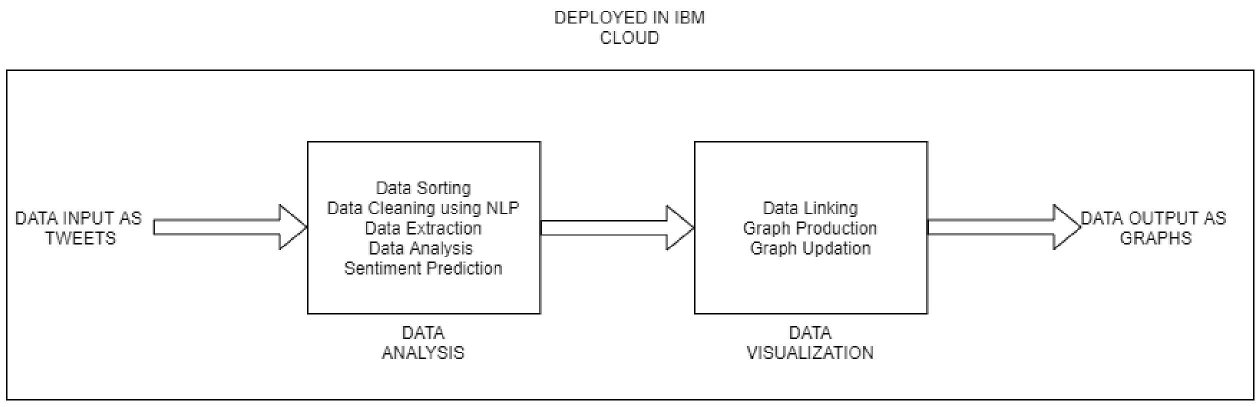
2.2 Solution

The solution to massive data gathering and analysis comes in the form of various data science techniques and machine learning algorithms which can work on TerraByte range data within seconds. The proposed solution deals with the various phases in sentiment analysis using algorithms and hence minimal human intervention. Computer systems are capable of storing and processing large volumes very easily compared to a human being.

- The tweets made by Indians during the Lockdown days were collected from Twitter, these tweets were split into different groups corresponding to each Lockdown Phase.
- These tweets were cleaned using Natural Language Processing Tools and relevant Informations were extracted from it.
- The extracted Information includes trending topics,trending hashtags and also the sentiments included in each of the tweets.
- Some of the tweets were analyzed using IBM Tone Analyzer to extract the relevant tone from the tweet. (The tones include happy, sad, analytical, confident, joy etc.)
- The extracted data was used to create two ML Models. One for identifying the sentiment, if a new tweet was given as the input. The next will predict the sentiment graph of the future, if the lockdown was to be extended.
- Completely processed data will be used to create visualization dashboards with the help of dash using plotly graphs and pandas dataframe.
- The developed application will be deployed using IBM cloud.

3 Theoretical Analysis

3.1 Block Diagram



3.2 Hardware/ Software design

- Hardware Design
  - The application does not have any minimum hardware requirements.

- Software Design
  - Python 3.x: (*Programming language used)*
  - GetOldTweets: *(For scraping Tweets)*
  - Pandas: *(Used for analysing Tweets)*
  - NLTK (Natural Language Toolkit) and RE (Regular Expression): *(Used for processing Tweets)*
  - IBM Watson Tone Analyzer: *(To label the Tweets)*
  - TextBlob: *(For sentiment analysis)*
  - Gensim: *(The LDA model in Gensim is used for topic modelling and predictive analysis)*
  - Matplotlib and Plotly: *(Used for data visualization)*
  - Sklearn and Statsmodels: *(For creating ML Models)*
  - Dash (With HTML, CSS and Javascript): *(Web Framework)*
  - IBM Cloud: *(For deployment)*
  - Git: *(For version control)*

4 Experimental Investigations

Data analysis of Covid-19 tweets show how data can be used to assess human emotions and reactions. Various machine learning algorithms when trained using relevant data can predict how the reactions will be in future. This data will provide an insight into how people view various events and topics.

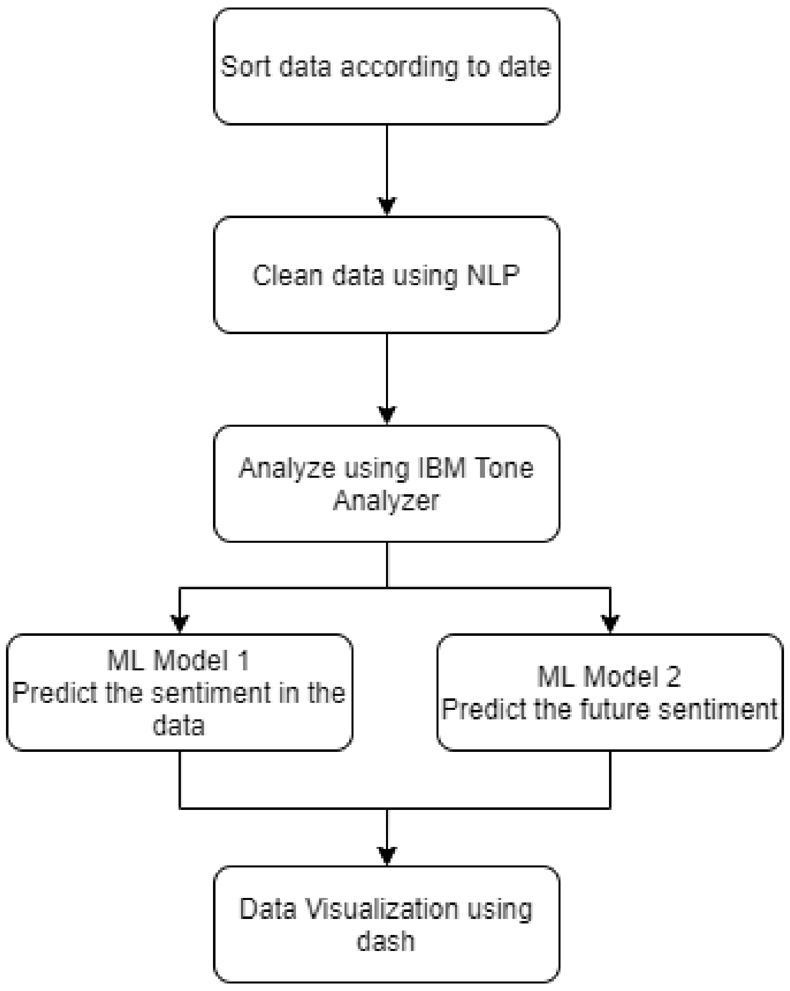
Steps Taken:

- Collected the tweets related to covid 19 made by indians during the lockdown period.
- By looking more into the data we found that many were expressing mixed reviews about lockdown extensions.
- Hypothesis: People supporting the notion thought that everything will subside in a few days and were hoping for the best, People against the notion thought that the lockdown will stagnate the economic growth of the country.
- Cleaned and processed the data
- Identified dominant topics,trending hashtags and sentiments of all the tweets.Visualisations were made using different graphs to convey our findings.
- The experiment supported our hypothesis and also gave some new Insights.
- The insights include: Almost half of the tweets were neutral, future trends were showing a decrease in positive sentiments if the lockdown was to increase, people were having many difficulties such as layoff from jobs, loss of income, decrease in mental health and so on.

Took time to learn about different technological fields such as data science, web development, machine learning,cloud computing and natural language processing.

Learned about different tools and technologies including IBM Cloud, Dash (Plotly), HTML/CSS, GitHub, Zoho Writer, Watson Tone Analyser, Pandas, NLTK and Sklearn.

5 Flowchart


Sort data according to date
Clean data using NLP
Analyze using IBM Tone Analyzer
ML Model 1 Predict the sentiment in the data
ML Model 2 Predict the future sentiment
Data Visualization using dash

6 Result

The desired visual representations of sentiment analysis of Covid-19 tweets has been obtained. A machine learning model which predicts the sentiment in tweets after the lockdown phase has been developed and deployed with 92% accuracy. A sentiment analyzer which takes text input and predicts the sentiment associated with it has been developed successfully. Covid-19 tweets have been extracted, processed, analysed, visualized and deployed successfully.

Also the shift in the Sentiment Graph, if the lockdown was to continue, was successfully identified using a machine learning model. A topic model was generated using LDA to identify the dominant topics discussed at that time.

7 Advantages and Disadvantages

- Advantages:
  - TeraByte range of data can be analysed in minimum time.
  - Requires minimal human intervention as the complete application runs on algorithms.
  - Ease of comparison of data using the various visualization tools.
  - Identification of sentiments and future trends.
  - Identification of trending topics in the given time period so that

- Disadvantages:
  - Computer programs have problems recognizing things like sarcasm and irony, negations, jokes, and exaggerations.
  - Negations are not identified (Example: "This is not funny" may be categorized as positive).

8 Applications

- **Decision Making:** By analysing the reaction of people to various decisions, the government can identify the decisions which the community supported and the decisions that were rejected, which will help in *implementing necessary changes* and also in *future decision making.*

- **Providing Assistance:** Sentiment analysis can also help in identifying the sections of the community that require any sort of help and as the data is collected real time, *assistance* can be provided immediately.

- **Prevention of fake news spread:** The analysis of trending topics can also help in detection of any *fake news* and hence prevent the spread of any misinformation.

- **Product recommendations:** By classifying tweets as Positive/Negative about a given product, we will be able to identify what product should be recommended to an individual and what should not.

9 Conclusion

In the era where data rules the world, sentiment analysis provides an easier way of interpreting reactions to various events, decisions, products etc. This provides a unique and easy method to assess crowd reaction. We have seen how sentiment analysis can help the government in decision making in times of a pandemic. We also implemented Machine Learning approaches to sentiment analysis. We have also seen the visual implementation of the analysed data and have gone through the applications offered by the project.

10 Future Scope

Organisations can use the insights learned from the visualization and analysis for research, decision making, quality improvement of services/products and detection of fake news spread. The data can also be used for recommending products/services to individuals. Government can analyse the public's reaction to various decisions being taken in the time of lockdown and thus to help in future decision making.

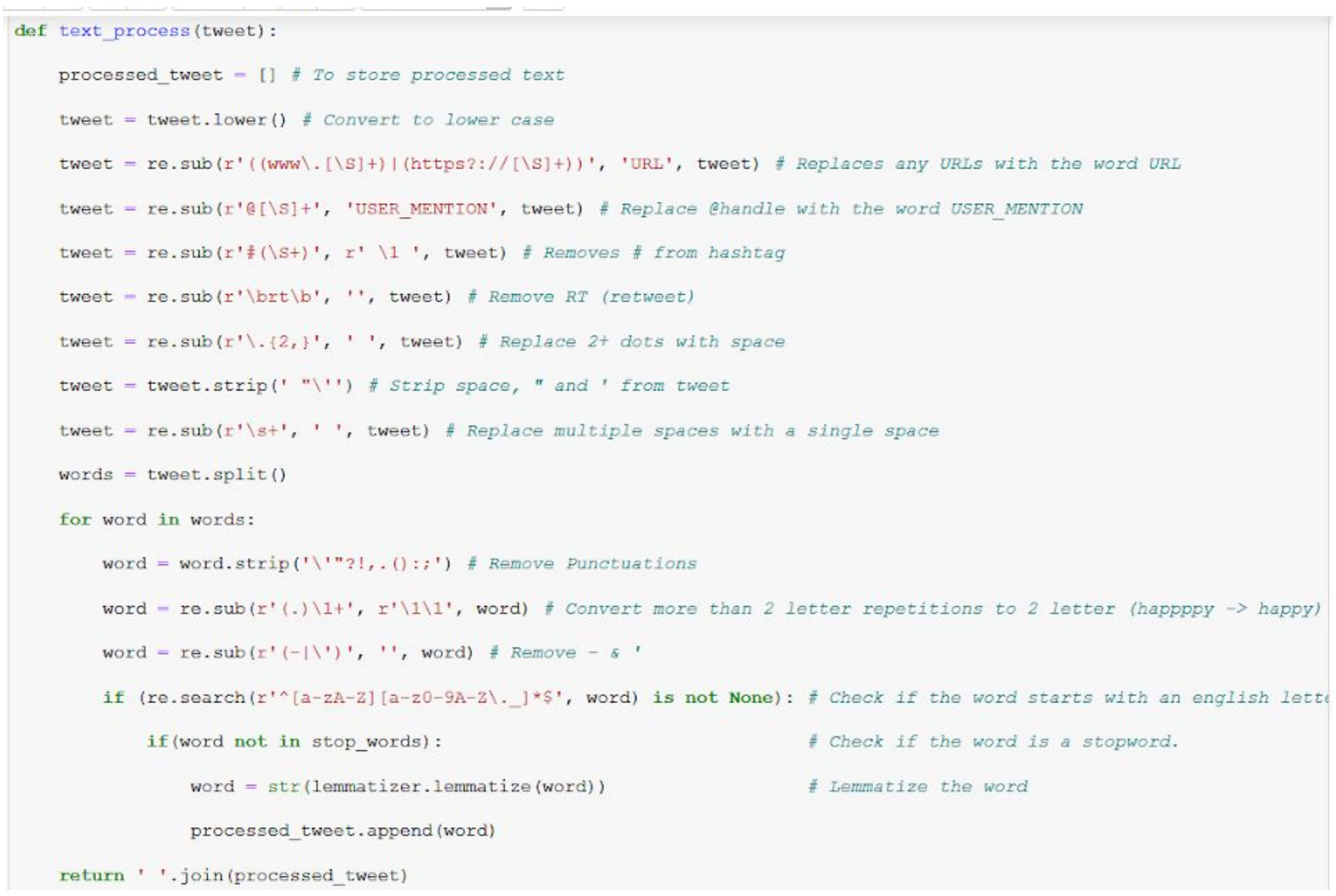
11 Bibliography

1. Charming Data: Youtube Channel - https://youtu.be/hSPmj7mK6ng
2. Plotly Graphs - https://plotly.com/python/
3. Dash Documentation - https://dash.plotly.com/
4. Data Science Blogs - https://towardsdatascience.com/data-science/home
5. Pandas Guide - https://pandas.pydata.org/docs/user_guide/index.html
6. Stackoverflow
7. GeeksforGeeks

Appendix

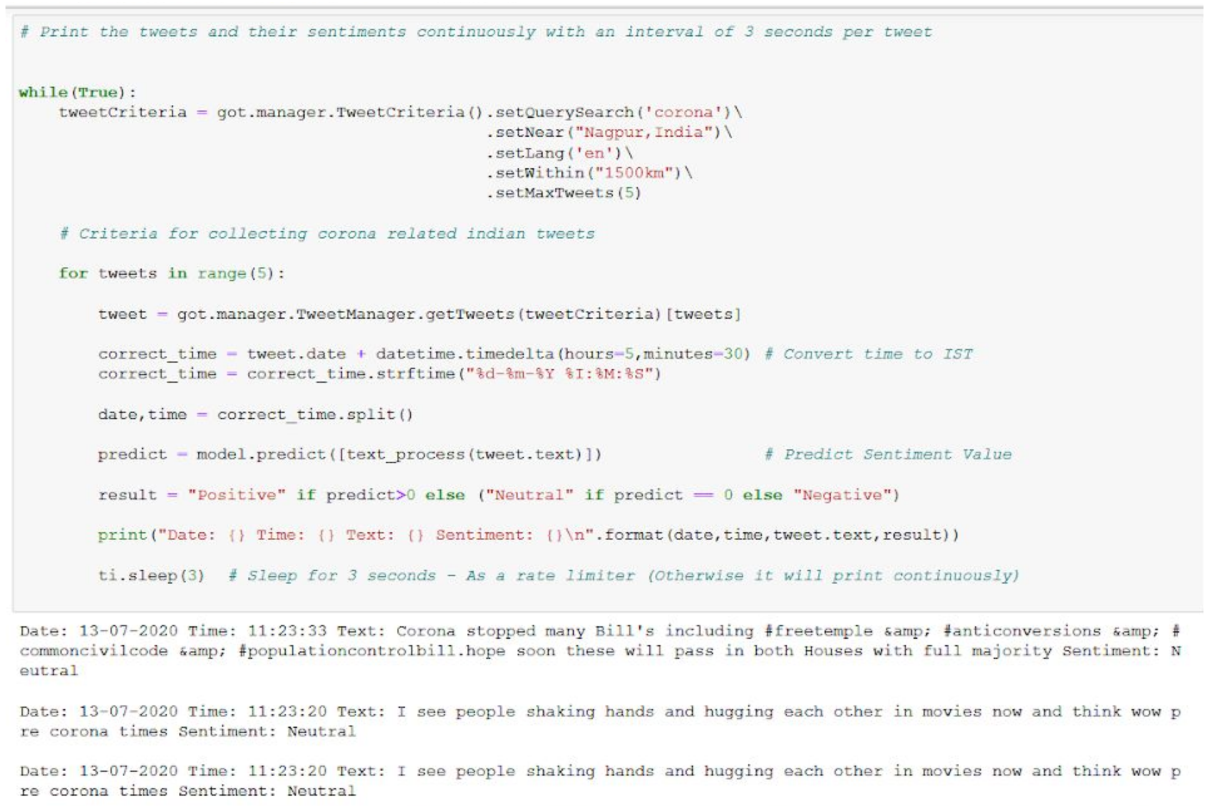
• Source Code

1. Clean Tweets



```
def text_process(tweet):

    processed_tweet = [] # To store processed text

    tweet = tweet.lower() # Convert to lower case

    tweet = re.sub(r'((www\.[\S]+)|(https?://[\S]+))', 'URL', tweet) # Replaces any URLs with the word URL

    tweet = re.sub(r'@[\S]+', 'USER_MENTION', tweet) # Replace @handle with the word USER_MENTION

    tweet = re.sub(r'#(\S+)', r' \1 ', tweet) # Removes # from hashtag

    tweet = re.sub(r'\brt\b', '', tweet) # Remove RT (retweet)

    tweet = re.sub(r'\.{2,}', ' ', tweet) # Replace 2+ dots with space

    tweet = tweet.strip(' "\'') # Strip space, " and ' from tweet

    tweet = re.sub(r'\s+', ' ', tweet) # Replace multiple spaces with a single space

    words = tweet.split()

    for word in words:

        word = word.strip('\'"?!,.():;') # Remove Punctuations

        word = re.sub(r'(.)\1+', r'\1\1', word) # Convert more than 2 letter repetitions to 2 letter (happppy -> happy)

        word = re.sub(r'(-|\')', '', word) # Remove - & '

        if (re.search(r'^[a-zA-Z][a-z0-9A-Z\._]*$', word) is not None): # Check if the word starts with an english lett

            if(word not in stop_words):                                 # Check if the word is a stopword.

                word = str(lemmatizer.lemmatize(word))                  # Lemmatize the word

                processed_tweet.append(word)

    return ' '.join(processed_tweet)
```

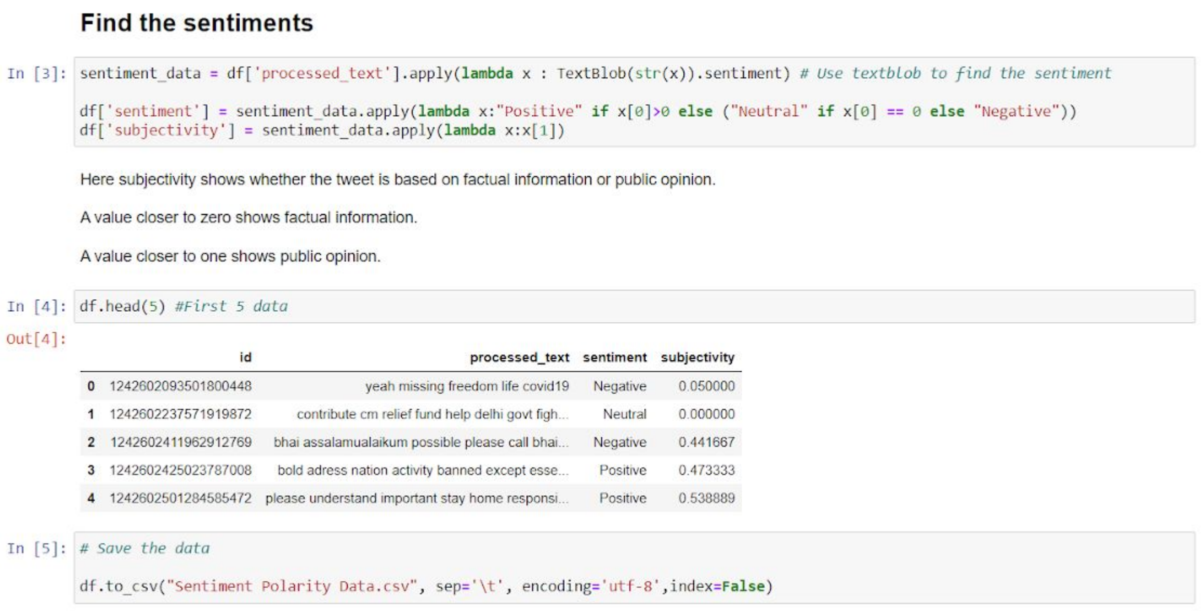
2. Live Tweets



```
# Print the tweets and their sentiments continuously with an interval of 3 seconds per tweet

while(True):
    tweetCriteria = got.manager.TweetCriteria().setQuerySearch('corona')\
                                            .setNear("Nagpur,India")\
                                            .setLang('en')\
                                            .setWithin("1500km")\
                                            .setMaxTweets(5)

    # Criteria for collecting corona related indian tweets

    for tweets in range(5):

        tweet = got.manager.TweetManager.getTweets(tweetCriteria)[tweets]

        correct_time = tweet.date + datetime.timedelta(hours=5,minutes=30) # Convert time to IST
        correct_time = correct_time.strftime("%d-%m-%Y %I:%M:%S")

        date,time = correct_time.split()

        predict = model.predict([text_process(tweet.text)])                # Predict Sentiment Value

        result = "Positive" if predict>0 else ("Neutral" if predict == 0 else "Negative")

        print("Date: {} Time: {} Text: {} Sentiment: {}\n".format(date,time,tweet.text,result))

        ti.sleep(3)  # Sleep for 3 seconds - As a rate limiter (Otherwise it will print continuously)
```

```
Date: 13-07-2020 Time: 11:23:33 Text: Corona stopped many Bill's including #freetemple &amp; #anticonversions &amp; #
commoncivilcode &amp; #populationcontrolbill.hope soon these will pass in both Houses with full majority Sentiment: N
eutral

Date: 13-07-2020 Time: 11:23:20 Text: I see people shaking hands and hugging each other in movies now and think wow p
re corona times Sentiment: Neutral

Date: 13-07-2020 Time: 11:23:20 Text: I see people shaking hands and hugging each other in movies now and think wow p
re corona times Sentiment: Neutral
```

3. Sentiment Analysis



Find the sentiments

```
In [3]: sentiment_data = df['processed_text'].apply(lambda x : TextBlob(str(x)).sentiment) # Use textblob to find the sentiment

        df['sentiment'] = sentiment_data.apply(lambda x:"Positive" if x[0]>0 else ("Neutral" if x[0] == 0 else "Negative"))
        df['subjectivity'] = sentiment_data.apply(lambda x:x[1])
```

Here subjectivity shows whether the tweet is based on factual information or public opinion.

A value closer to zero shows factual information.

A value closer to one shows public opinion.

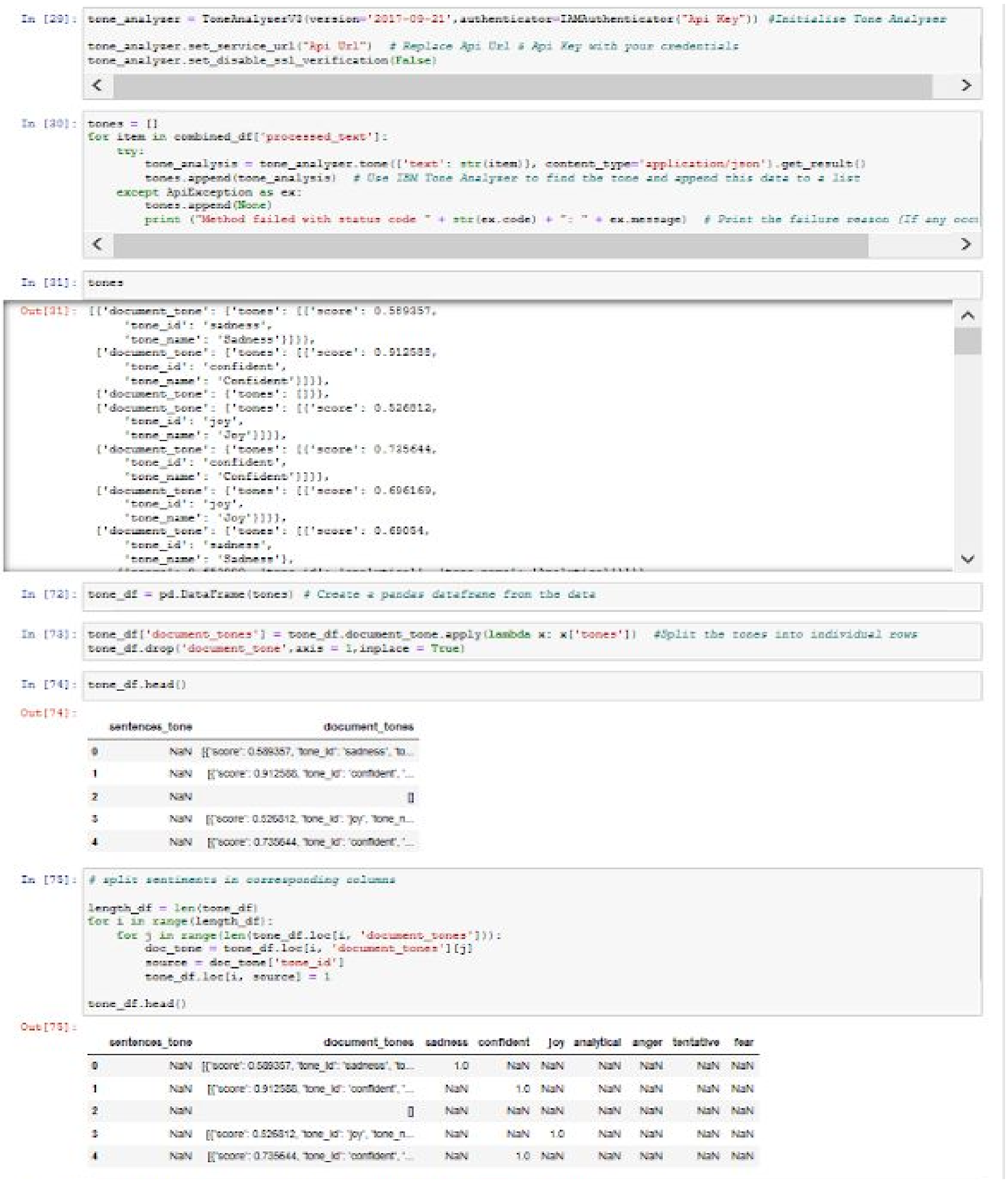
```
In [4]: df.head(5) #First 5 data
```

Out[4]:

	id	processed_text	sentiment	subjectivity
0	1242602093501800448	yeah missing freedom life covid19	Negative	0.050000
1	1242602237571919872	contribute cm relief fund help delhi govt figh...	Neutral	0.000000
2	1242602411962912769	bhai assalamualaikum possible please call bhai...	Negative	0.441667
3	1242602425023787008	bold adress nation activity banned except esse...	Positive	0.473333
4	1242602501284585472	please understand important stay home responsi...	Positive	0.538889

```
In [5]: # Save the data

        df.to_csv("Sentiment Polarity Data.csv", sep='\t', encoding='utf-8',index=False)
```

4. Tone Analyser



```
In [29]: tone_analyser = ToneAnalyzerV3(version='2017-09-21',authenticator=IAMAuthenticator("Api Key")) #Initialise Tone Analyser

         tone_analyser.set_service_url("Api Url")  # Replace Api Url & Api Key with your credentials
         tone_analyser.set_disable_ssl_verification(False)
```

```
In [30]: tones = []
         for item in combined_df['processed_text']:
             try:
                 tone_analysis = tone_analyser.tone({'text': str(item)}, content_type='application/json').get_result()
                 tones.append(tone_analysis)  # Use IBM Tone Analyzer to find the tone and append this data to a list
             except ApiException as ex:
                 tones.append(None)
                 print ("Method failed with status code " + str(ex.code) + ": " + ex.message)  # Print the failure reason (If any occu
```

```
In [31]: tones
```

```
Out[31]: [{'document_tone': {'tones': [{'score': 0.589357,
              'tone_id': 'sadness',
              'tone_name': 'Sadness'}]}},
           {'document_tone': {'tones': [{'score': 0.912588,
              'tone_id': 'confident',
              'tone_name': 'Confident'}]}},
           {'document_tone': {'tones': []}},
           {'document_tone': {'tones': [{'score': 0.526812,
              'tone_id': 'joy',
              'tone_name': 'Joy'}]}},
           {'document_tone': {'tones': [{'score': 0.735644,
              'tone_id': 'confident',
              'tone_name': 'Confident'}]}},
           {'document_tone': {'tones': [{'score': 0.696169,
              'tone_id': 'joy',
              'tone_name': 'Joy'}]}},
           {'document_tone': {'tones': [{'score': 0.69084,
              'tone_id': 'sadness',
              'tone_name': 'Sadness'},
```

```
In [72]: tone_df = pd.DataFrame(tones) # Create a pandas dataframe from the data
```

```
In [73]: tone_df['document_tones'] = tone_df.document_tone.apply(lambda x: x['tones'])  #Split the tones into individual rows
         tone_df.drop('document_tone',axis = 1,inplace = True)
```

```
In [74]: tone_df.head()
```

Out[74]:

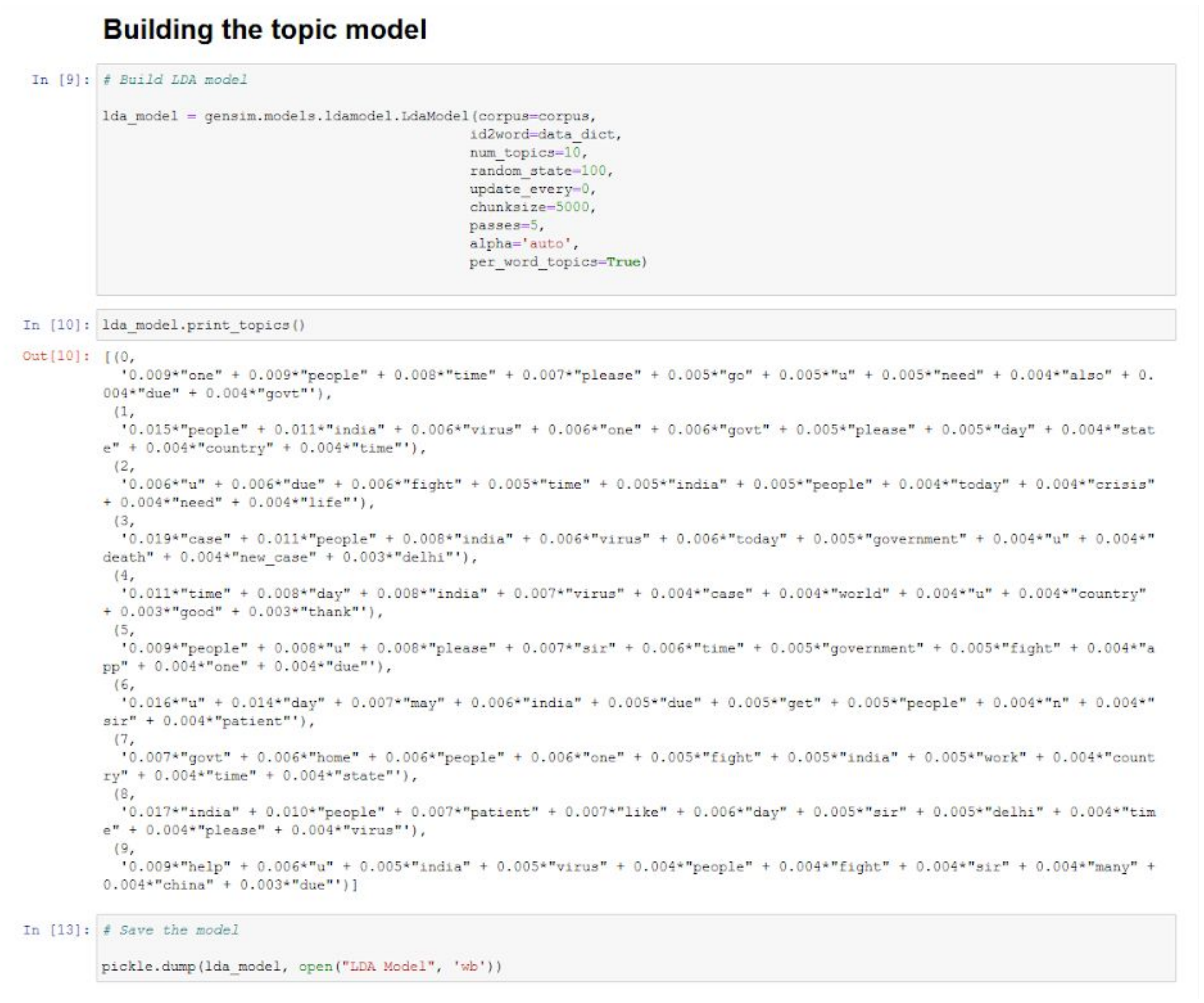
	sentences_tone	document_tones
0	NaN	[{'score': 0.589357, 'tone_id': 'sadness', 'to...
1	NaN	[{'score': 0.912588, 'tone_id': 'confident', '...
2	NaN	[]
3	NaN	[{'score': 0.526812, 'tone_id': 'joy', 'tone_n...
4	NaN	[{'score': 0.735644, 'tone_id': 'confident', '...

```
In [75]: # split sentiments in corresponding columns

         length_df = len(tone_df)
         for i in range(length_df):
             for j in range(len(tone_df.loc[i, 'document_tones'])):
                 doc_tone = tone_df.loc[i, 'document_tones'][j]
                 source = doc_tone['tone_id']
                 tone_df.loc[i, source] = 1

         tone_df.head()
```

Out[75]:

	sentences_tone	document_tones	sadness	confident	joy	analytical	anger	tentative	fear
0	NaN	[{'score': 0.589357, 'tone_id': 'sadness', 'to...	1.0	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	[{'score': 0.912588, 'tone_id': 'confident', '...	NaN	1.0	NaN	NaN	NaN	NaN	NaN
2	NaN	[]	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	[{'score': 0.526812, 'tone_id': 'joy', 'tone_n...	NaN	NaN	1.0	NaN	NaN	NaN	NaN
4	NaN	[{'score': 0.735644, 'tone_id': 'confident', '...	NaN	1.0	NaN	NaN	NaN	NaN	NaN

5. Topic Model



Building the topic model

```
In [9]:  # Build LDA model

         lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                                    id2word=data_dict,
                                                    num_topics=10,
                                                    random_state=100,
                                                    update_every=0,
                                                    chunksize=5000,
                                                    passes=5,
                                                    alpha='auto',
                                                    per_word_topics=True)
```

```
In [10]: lda_model.print_topics()
```

```
Out[10]: [(0,
           '0.009*"one" + 0.009*"people" + 0.008*"time" + 0.007*"please" + 0.005*"go" + 0.005*"u" + 0.005*"need" + 0.004*"also" + 0.
          004*"due" + 0.004*"govt"'),
          (1,
           '0.015*"people" + 0.011*"india" + 0.006*"virus" + 0.006*"one" + 0.006*"govt" + 0.005*"please" + 0.005*"day" + 0.004*"stat
          e" + 0.004*"country" + 0.004*"time"'),
          (2,
           '0.006*"u" + 0.006*"due" + 0.006*"fight" + 0.005*"time" + 0.005*"india" + 0.005*"people" + 0.004*"today" + 0.004*"crisis"
          + 0.004*"need" + 0.004*"life"'),
          (3,
           '0.019*"case" + 0.011*"people" + 0.008*"india" + 0.006*"virus" + 0.006*"today" + 0.005*"government" + 0.004*"u" + 0.004*"
          death" + 0.004*"new_case" + 0.003*"delhi"'),
          (4,
           '0.011*"time" + 0.008*"day" + 0.008*"india" + 0.007*"virus" + 0.004*"case" + 0.004*"world" + 0.004*"u" + 0.004*"country"
          + 0.003*"good" + 0.003*"thank"'),
          (5,
           '0.009*"people" + 0.008*"u" + 0.008*"please" + 0.007*"sir" + 0.006*"time" + 0.005*"government" + 0.005*"fight" + 0.004*"a
          pp" + 0.004*"one" + 0.004*"due"'),
          (6,
           '0.016*"u" + 0.014*"day" + 0.007*"may" + 0.006*"india" + 0.005*"due" + 0.005*"get" + 0.005*"people" + 0.004*"n" + 0.004*"
          sir" + 0.004*"patient"'),
          (7,
           '0.007*"govt" + 0.006*"home" + 0.006*"people" + 0.006*"one" + 0.005*"fight" + 0.005*"india" + 0.005*"work" + 0.004*"count
          ry" + 0.004*"time" + 0.004*"state"'),
          (8,
           '0.017*"india" + 0.010*"people" + 0.007*"patient" + 0.007*"like" + 0.006*"day" + 0.005*"sir" + 0.005*"delhi" + 0.004*"tim
          e" + 0.004*"please" + 0.004*"virus"'),
          (9,
           '0.009*"help" + 0.006*"u" + 0.005*"india" + 0.005*"virus" + 0.004*"people" + 0.004*"fight" + 0.004*"sir" + 0.004*"many" +
          0.004*"china" + 0.003*"due"')]
```

```
In [13]: # Save the model

         pickle.dump(lda_model, open("LDA Model", 'wb'))
```

6. Dash App Code - With rest of the prediction and visualisation

```
import pandas as pd

import plotly

import plotly.express as px  # (version 4.7.0)
import plotly.graph_objects as go


import dash  # (version 1.12.0) pip install dash

import dash_core_components as dcc

import dash_html_components as html

from dash.dependencies import Input, Output


import GetOldTweets3 as got


from statsmodels.tsa.arima_model import ARIMA


from datetime import datetime as dt, timedelta

import time as ti

import pickle

import nltk

import re

from nltk.stem import WordNetLemmatizer

from nltk.corpus import stopwords


stop_words = set(stopwords.words('english'))  # Set of stopwords (Words
that doesn't give meaningful information)
```

```
lemmatizer = WordNetLemmatizer()  # Used for converting words with similar
meaning to a single word.

def text_process(tweet):

    processed_tweet = [] # To store processed text

    tweet = tweet.lower() # Convert to lower case

      tweet  =  re.sub(r'((www\.[\S]+)|(https?://[\S]+))',  'URL',  tweet)  #
Replaces any URLs with the word URL

      tweet  =  re.sub(r'@[\S]+',  'USER_MENTION',  tweet)  #  Replace  @handle
with the word USER_MENTION

    tweet = re.sub(r'#(\S+)', r' \1 ', tweet) # Removes # from hashtag

    tweet = re.sub(r'\brt\b', '', tweet) # Remove RT (retweet)

    tweet = re.sub(r'\.{2,}', ' ', tweet) # Replace 2+ dots with space

    tweet = tweet.strip(' "\'') # Strip space, " and ' from tweet
    tweet = re.sub(r'\s+', ' ', tweet) # Replace multiple spaces with a
single space

    words = tweet.split()

    for word in words:

        word = word.strip('\'"?!,.():;') # Remove Punctuations

        word = re.sub(r'(.)\1+', r'\1\1', word) # Convert more than 2
letter repetitions to 2 letter (happppy -> happy)

        word = re.sub(r'(-|\')', '', word) # Remove - & '

        if (re.search(r'^[a-zA-Z][a-z0-9A-Z\._]*$', word) is not None): #
Check if the word starts with an english letter

            if(word not in stop_words):                              #
Check if the word is a stopword.

                word = str(lemmatizer.lemmatize(word))               #
Lemmatize the word

                processed_tweet.append(word)
    return ' '.join(processed_tweet)


predictor = pickle.load(open("Sentiment Predictor", 'rb'))  # Code to load
ML model for later use


app = dash.Dash(__name__)


date_range = {1:['2020-03-25','2020-04-14'],

              2:['2020-04-15','2020-05-03'],

              3:['2020-05-04','2020-05-17'],

              4:['2020-05-18','2020-05-31'],

              5:['2020-06-01','2020-06-14']}


phase_range = {1:'LD1',2:'LD2',3:'LD3',4:'LD4',5:'Unlock1'}


#
---------------------------------------------------------------------------
--------------------------------------------------------------

#Import and clean data (importing csv into pandas)

df = pd.read_csv("Hashtag data.csv")

df1 = pd.read_csv("Topic Data.csv")

df2 = pd.read_csv("Res.csv")
```

```
df2['Date'] = pd.to_datetime(df2['Date'])

df3 = pd.read_csv("Tone Data.csv")

df4 = pd.read_csv("Graph Prediction.csv")




#
---------------------------------------------------------------------------
--------------------------------------------------------------

# App layout

app.layout = html.Div([



    html.H1("COVID-19 Sentiment Analysis",className = "jumbotron"),



    dcc.Slider(id="slct_period",

                min=0,

                max=5,

                marks={

                    0:'General',

                    1:'LD1',

                    2:'LD2',

                    3:'LD3',

                    4:'LD4',
                    5:'Unlock1'

                },

                value=0

                ),


    html.Div(

    dcc.Graph(id = 'sentiment_analysis2')),


    html.Div(

    dcc.Graph(id = 'sentiment_analysis'),

       style={'padding-left':'80px','padding-right':'50px','width':  '40%',
'display': 'inline-block'}),


    html.Div(

    dcc.Graph(id = 'sentiment_analysis1'),

            style={'padding-top':   '25px',   'width':   '40%',   'display':
'inline-block','padding-left':'50px','padding-right':'50px'}),


    html.Div(

    dcc.Graph(id = 'sentiment_analysis3'),

        style={'width':  '50%',  'display':  'inline-block','padding-top':
'50px','padding-bottom': '50px'}),
    html.Div(dcc.Graph(id='sentiment_graph')),



       html.H3("Sentiment  Prediction  end  date:  ",style={'width':  '50%',
'display': 'inline-block', 'text-align': 'right','padding-right':'10px'}),



    html.Div(

    dcc.DatePickerSingle(

        id='sentiment_date',

        min_date_allowed=dt(2020, 6, 1),

        max_date_allowed=dt(2020, 9, 1),

        initial_visible_month=dt(2020,6,1),

        date='2020-6-2'),

                  style={'margin-left':'-5px','width':  '40%',  'display':
'inline-block'}),



    html.Div(

              html.Button('Live    Tweet    Sentiments',    id='submit-val',
n_clicks=0,className = 'livebutton'),

    className = 'livecenter'),



    html.Div(id='sentiment_live'),



    html.Div(
    dcc.Input(

            id="sentiment_text",

            type = 'text',

            value='',

            placeholder="Input the text")),


    html.Div(id="sentiment_prediction")



    ])




#
---------------------------------------------------------------------------
----

# Connect the Plotly graphs with Dash Components

@app.callback(

     Output('sentiment_analysis', 'figure'),

    [Input('slct_period', 'value')]

)


def update_graph(slct_period):

    if slct_period == 0:
```

```
                                                            df_plot        =
df.copy().groupby('hashtag').sum().reset_index().sort_values(by='value',as
cending=False)

    else:

        df_plot = df[df['Phase']==phase_range[slct_period]]


                                                                fig              =
px.bar(df_plot,x='hashtag',y='value',color='value',labels={'hashtag':'Hash
tags','value':"Values"},height=600,width=600)

                            fig.update_layout(title={'text':       "Trending
Hashtags",'y':0.95,'x':0.47,'xanchor': 'center','yanchor': 'top'})


    return fig


@app.callback(

     Output('sentiment_analysis1', 'figure'),

    [Input('slct_period', 'value')]

)


def update_graph(slct_period):

    if slct_period == 0:

                                                            df_plot        =
df1.copy().groupby('topic_data').sum().reset_index().sort_values(by='topic
_count',ascending=False)

    else:
        df_plot = df1[df1['Phase']==phase_range[slct_period]]

                                                         fig =
px.bar(df_plot,x='topic_data',y='topic_count',color='topic_count',labels={
'topic_data':'Topics','topic_count':"Values"},height=600,width=600)

         fig.update_layout(xaxis=dict(showticklabels=False),title={'text':
"Trending Topics",'y':0.95,'x':0.49,'xanchor': 'center','yanchor': 'top'})



    return fig



@app.callback(

     Output('sentiment_analysis2', 'figure'),

    [Input('slct_period', 'value')]

)



def update_graph(slct_period):

    if slct_period == 0:

        df_plot = df2.copy()

    else:

        start,end=map(str,date_range[slct_period])

        df_plot = df2[(df2['Date'] >= start) & (df2['Date'] <= end)]

    fig = px.line(df_plot,x='Date',y='Value',color='Sentiment')

                    fig.update_layout(title={'text': "Sentiment of
People",'y':0.95,'x':0.48,'xanchor': 'center','yanchor': 'top'})
```

```
    return fig


@app.callback(
     Output('sentiment_analysis3', 'figure'),
    [Input('slct_period', 'value')]
)


def update_graph(slct_period):
    if slct_period == 0:
                                                                       df_plot        =
df3.copy().groupby('Tone').sum().reset_index().sort_values(by='Value',asce
nding=False)
    else:
        start,end=map(str,date_range[slct_period])
        df_plot = df3[df3['Phase']==phase_range[slct_period]]
    fig = px.pie(df_plot,names="Tone",values="Value")
             fig.update_layout(title={'text':   "Tone   of   500   Sample
tweets",'y':0.95,'x':0.47,'xanchor': 'center','yanchor': 'top'})


    return fig


@app.callback(
     Output('sentiment_live', 'children'),

    [Input('submit-val', 'n_clicks')]

)



def update_graph(slct_period):

     tweetCriteria = got.manager.TweetCriteria().setQuerySearch('corona OR
coronavirus OR covid-19 OR covid19 OR covid OR pandemic OR lockdown')\

                                               .setNear("Nagpur,India")\

                                               .setLang('en')\

                                               .setWithin("1500km")\

                                               .setMaxTweets(1)



    tweet = got.manager.TweetManager.getTweets(tweetCriteria)[0]

    correct_time = tweet.date + timedelta(hours=5,minutes=30)

    correct_time = correct_time.strftime("%d-%m-%Y %I:%M:%S")

    date,time = correct_time.split()

    predict = predictor.predict([text_process(tweet.text)])

     result = "Positive" if predict>0 else ("Neutral" if predict == 0 else
"Negative")



           return   ("   Date:   {}   ".format(date),html.Br(),"   Time:   {}
".format(time),html.Br(),"    Text:    {}    ".format(tweet.text),html.Br(),"
Sentiment: {} ".format(result))
```

```
@app.callback(
    Output('sentiment_graph', 'figure'),
    [Input('sentiment_date', 'date')])

def update_output(date):

    start = dt(2020,6,1)
    end = date
    year, month, day = map(int, end.split('-'))
    end = dt(year, month, day)

    result = df4
    train = result.iloc[:68]
    test = result.iloc[68:]
    model = ARIMA(train.Positive, order=(2,2,1))
    model_fit = model.fit(disp=-1)

    delta = end - start
    test_dates = []
    for i in range(delta.days + 1):
        test_dates.append(str(start + timedelta(days=i)).split()[0])
    forecast = model_fit.forecast(steps=delta.days+1)

                                                        predicted_data              =
pd.DataFrame(forecast[0],index=test_dates,columns=['Positive'])

    #Neutral
    model = ARIMA(train.Neutral, order=(2,2,1))
    model_fit = model.fit(disp=-1)
    forecast = model_fit.forecast(steps=delta.days+1)
    predicted_data['Neutral'] = forecast[0]

    #Negative
    model = ARIMA(train.Negative, order=(2,2,1))
    model_fit = model.fit(disp=-1)
    forecast = model_fit.forecast(steps=delta.days+1)
    predicted_data['Negative'] = forecast[0]

    #Plot the data

                                                                 fig                =
px.line(predicted_data,labels={'index':'Date','value':"Percentage   Change
in Sentiments"})
```

```
            fig.update_layout(legend_title_text='Sentiment',title={'text':
"Predicted          Sentiment          Graph",'y':0.95,'x':0.47,'xanchor':
'center','yanchor': 'top'})


    return fig



@app.callback(

     Output('sentiment_prediction', 'children'),

    [Input('sentiment_text', 'value')]

)


def update_output_div(input_text):


    predict = predictor.predict([text_process(input_text)])


     result = "Positive" if predict>0 else ("Neutral" if predict == 0 else
"Negative")


    return 'Sentiment is : {}'.format(result)
```

```
#
--------------------------------------------------------------------------
----

if __name__ == '__main__':

    app.run_server(debug=True)
```