MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥
Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

B. Tech Trimester X MINI PROJECT REPORT

On

COVID-19 SENTIMENT ANALYSIS AND DASHBOARDING

Submitted by,

DEVANSH MISHRA (1032170933)

Project Guide:

Dr. Apurva Naik

Year: 2020-2021

School of Electronics and Communication Engineering
MIT World Peace University, Pune

MIT-WPU

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

School of Electronics and Communication Engineering

MIT World Peace University

Pune

CERTIFICATE

This is to certify that the B. Tech Mini Project entitled

COVID-19 SENTIMENT ANALYSIS AND DASHBOARDING

work has been carried out successfully by

Devansh Mishra (1032170933)

during the Academic Year 2020-2021 in partial fulfilment of their course of Mini Project for Final Year Electronics and Communication Engineering as per the guidelines prescribed by the MIT World Peace University, Pune

Project Guide	Bath Coordinator	Mini Project Coordinator
Dr. Apurva Naik	**Mrs. Aarti Vagga**	**Dr. Trushita S. Chaware**

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher and guide Dr. Apurva Naik for her able guidance and support in completing my project. I would also like to extend my gratitude to my Mini Project Coordinator Dr. Trushita S. Chaware for providing me with all the facility that was required.

I would also like to thank my parents and classmates who have helped me with their valuable suggestions and guidance which proved to be greatly beneficial in various phases of my project and were always present with their support and encouragement.

INDEX

CHAPTER -1

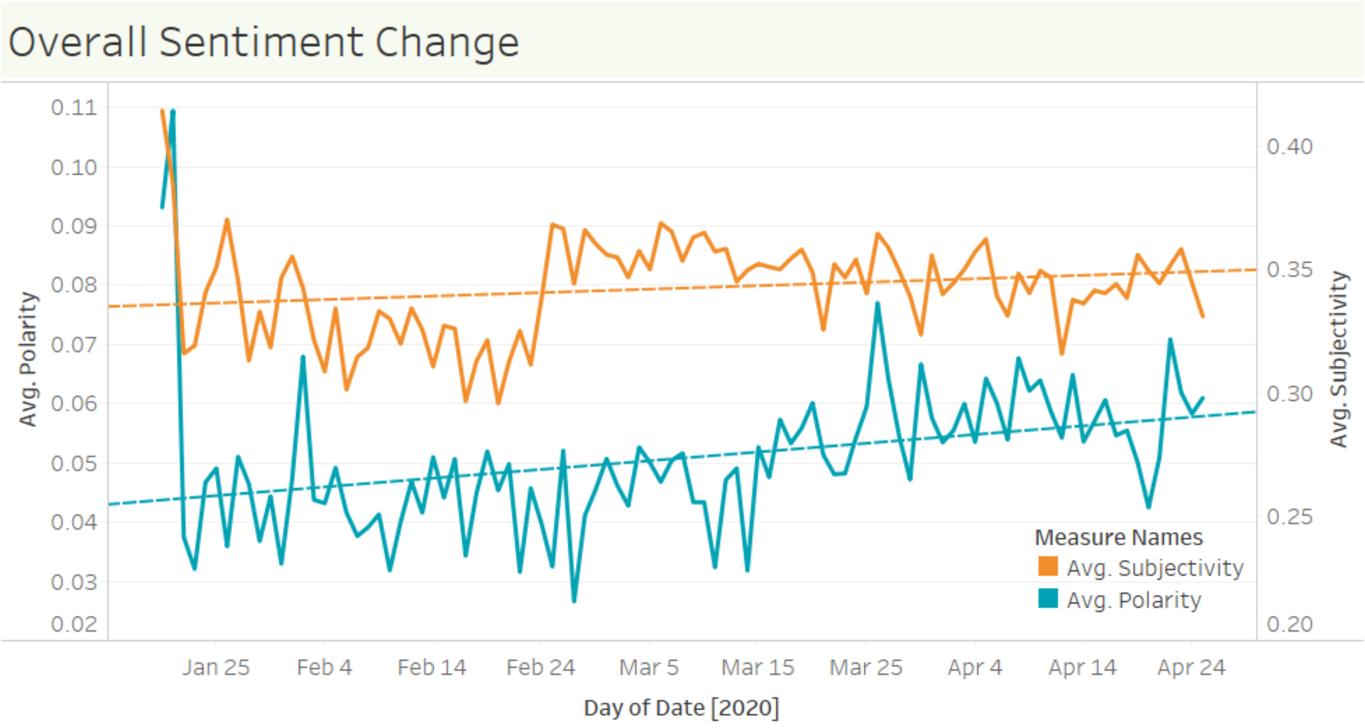
INTRODUCTION

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.

To study public sentiments, we chose Twitter as our target field. As one of the world's biggest social network platforms, Twitter hosts abundant user-generated posts, which closely reflect the public's reactions towards this pandemic with low latency. By deploying Natural Language Processing (NLP) methods on it, we were able to extract and quantify the public sentiments over time. The tool we used are TextBlob. We used TextBlob to explore public sentiments, which showed an upward trend in being steadily more positive.

Twitter is not only a place for people to respond to others' tweets but also a platform to post your tweets and share your feelings. Thus, besides likes/replies/retweets, we also mined the content of COVID-19 related tweets to see how people's feelings and expressions changed over time. With the help of TextBlob, a sentiment analysis library in Python, we extracted how subjective/objective (subjectivity) the content is and whether the content is positive or negative (polarity) for each tweet. The figure for the average subjectivity and polarity is shown below.

SCOPE AND OBJECTIVE



Avg. Subjectivity Trend Line
P-value: 0.0458903
Equation: Avg. Subjectivity = 0.000140279*Day of Date + -5.81566
Coefficients

Term	Value	StdErr	t-value	p-value
Day of Date	0.0001403	6.934e-05	2.02293	0.0458903
intercept	-5.81566	3.04408	-1.91048	0.0590872

Avg. Polarity Trend Line
P-value: 0.0007462
Equation: Avg. Polarity = 0.000147344*Day of Date + -6.41741
Coefficients

Term	Value	StdErr	t-value	p-value
Day of Date	0.0001473	4.228e-05	3.48509	0.0007462
intercept	-6.41741	1.85594	-3.45778	0.0008166

According to the chart above, with the development of COVID-19, the related tweets’ expression became more subjective (from about 0.33 to about 0.35) on average, and people’s feelings became more positive (from about 0.04 to about 0.06) on average. Why did this happen? Why with more and more people being infected with Coronavirus, the sentiment of related tweets went positive? With such questions, I went deep into what actual emotions the tweets reflected and what kinds of topics people talked about when mentioning this disease.

CHAPTER – 2

PRESENT SCENARIO

At the time of writing this document, there were more than 11 million confirmed cases of novel corona virus cases all over the world, and around 3M people are infected in USA alone. The number of total fatal cases exceeded 500k globally. The number of infected people, active cases, and fatality keep rising every day. The first confirmed case of novel corona virus disease was reported in Wuhan, China. However, over the last 5 months, the virus spread explosively all over the world. At the time of writing this document, the United States has the maximum number of corona virus cases and fatalities trailing by Brazil, and then followed by India.

Every country is taking preventive measurements to fight against the COVID-19 pandemic. Social distancing or stay at-home became the most widely used directive all over the world. Social distancing is forcing people to stay at home, and as a result, it is impacting the public event, business, education, and almost every other activity associated with the human life. People are also losing their jobs, and getting infected from corona and thus, stress is rising at the personal and at the community levels. Studies of behavioural economics show that emotions (Joy, Anger, Worry, Disgust, Fear, etc.) can deeply affect the individual behaviour and decision-making.

Social networks have the hidden potential to reveal valuable insights on human emotions at the personal and community level. Monitoring tweets could be valuable particularly during and after COVID-19 pandemic as the situation and people reaction both are changing every moment during this unpredictable time. Thus, the analysis of twitter data might play a crucial role to understand the people behaviour and response during the COVID-19 pandemic. Recent works

show that twitter data, and human emotions analysis can be useful for predicting crimes, stock market, election vote, disaster management, and more. Therefore, it is paramount to analyse the social media data to understand the human behaviour and reaction in the ongoing pandemic.

Twitter has been proven useful for various tasks such as emergency communication network, public emotions monitor, detect anomalies and provide early warning, etc. Twitter was used as the data source to monitor the public reaction and health during disasters (e.g. hurricanes, floods, earthquakes, terrorist bombing, public health related misinformation propagation and others), and disease outbreaks. Researchers are trying to come forward with various ideas which involves the use of twitter data in various ways.

We are continuously collecting the data since March 5, 2020 and will keep fetching the tweets using Twitter Streaming API2 and python Tweepy3 package. We have collected more than 200M tweets which has around 1.3TB of raw data until July 2nd, 2020 and saved this data as JSON files. We are using COVID-19 related keywords such as covid, corona for the tweets collection. The module listens the stream of the tweets and try to check if a tweet contains covid, or corona in it.

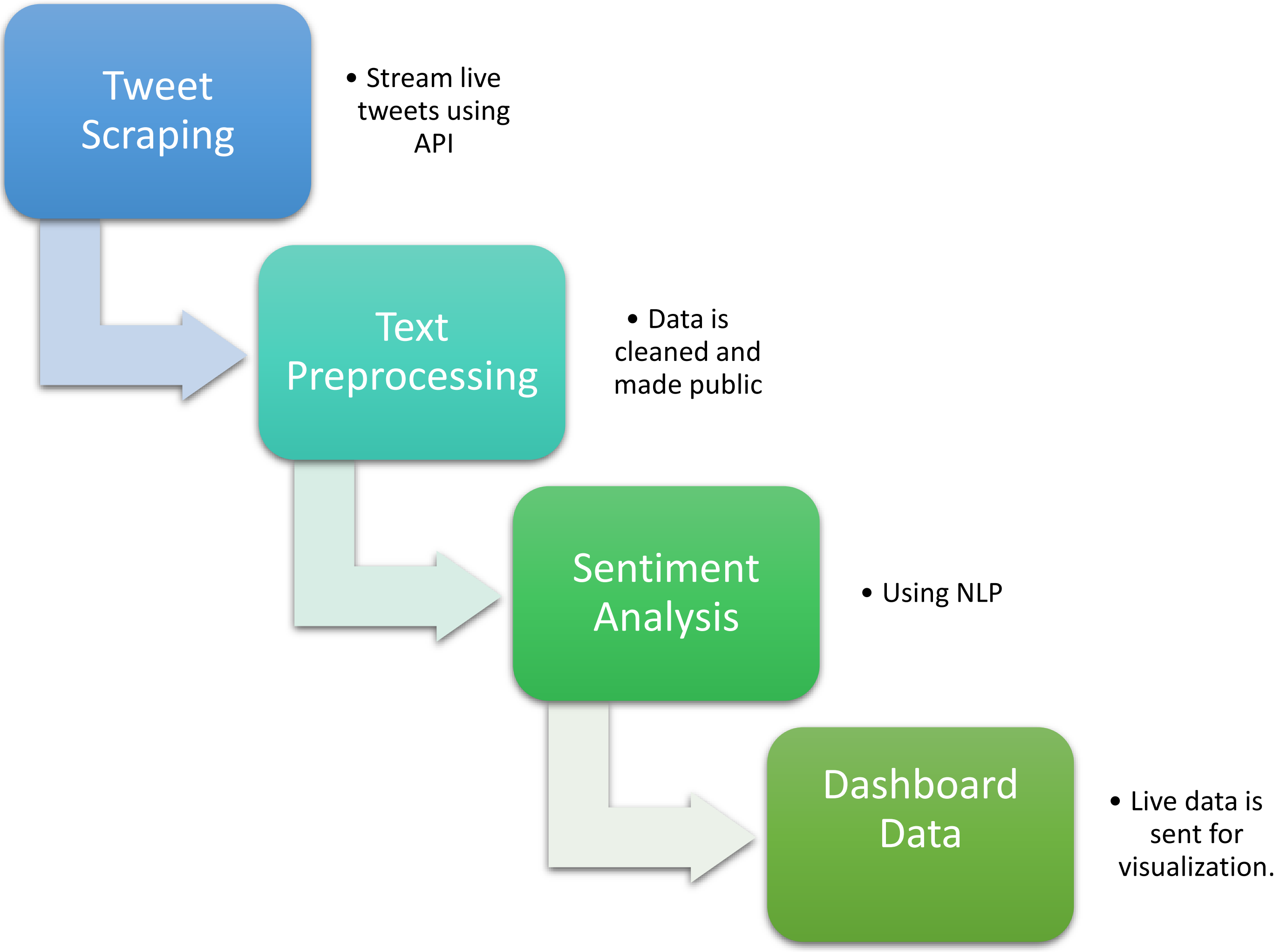
CHAPTER -3

SYSTEM SPECIFICATIONS

Listed below are the requirements essential for the functioning of the project:

- Python 3.x
- MySQL Community Edition
- Visual Studio Code
- Tweepy
- NumPy
- Pandas
- Seaborn
- Plotly
- Dash
- RE
- TextBlob
- NLTK
- Git
- Heroku

BLOCK DIAGRAM



Tweet Scraping

- Stream live tweets using API

Text Preprocessing

- Data is cleaned and made public

Sentiment Analysis

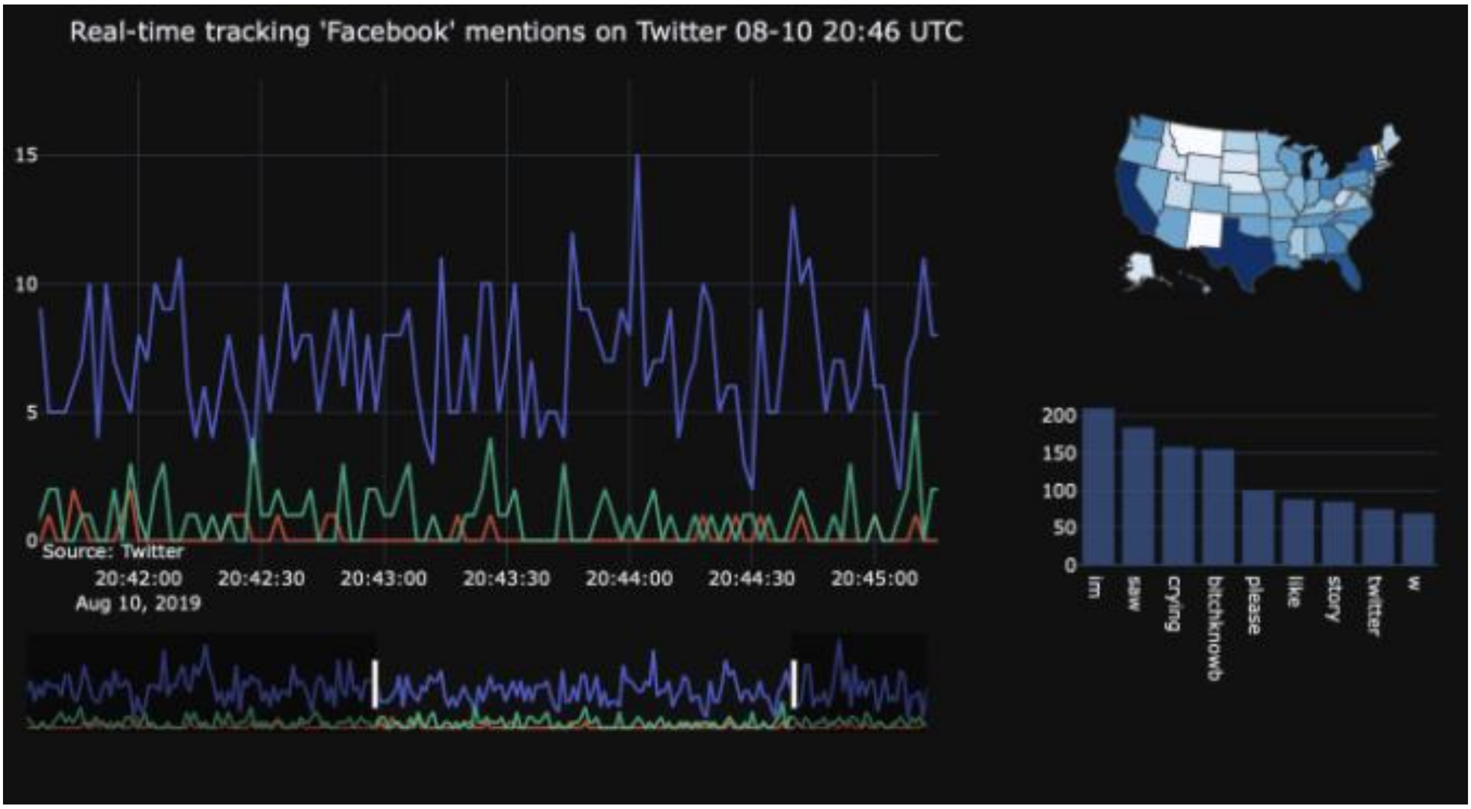
- Using NLP

Dashboard Data

- Live data is sent for visualization.

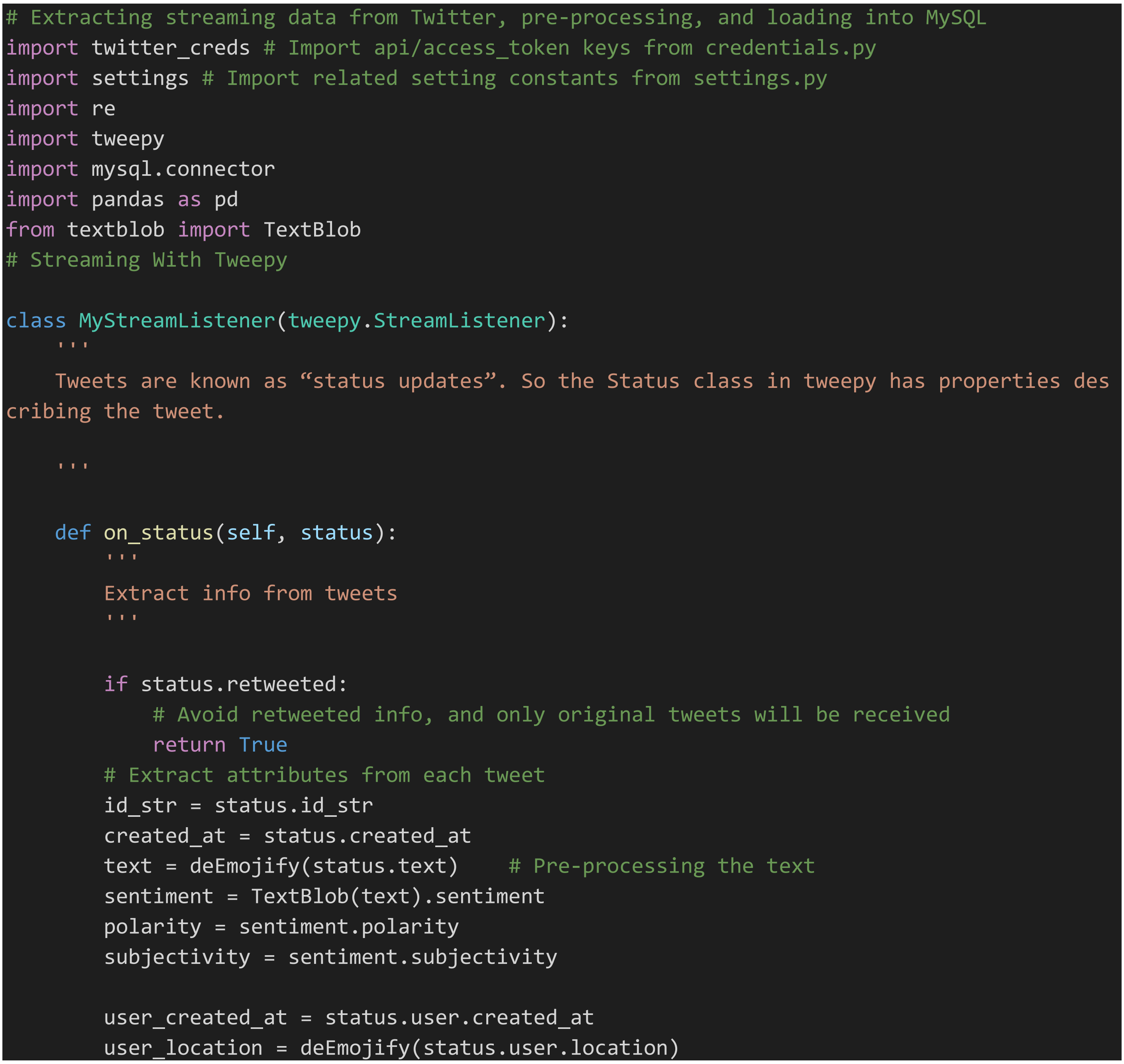
Data attributes

Feature	Description
tweet_id	Unique ID of a tweet.
created_at	Creation time of a tweet.
loc	State level user location.
text	Processed tweet text. All the text are in small letters, non-English characters and few stop words are removed.
user_id	Pseudo user id. The exact user name is transformed to a anonymous id to preserve the privacy of the user.
verified	Denotes whether the tweet post is verified or not (1 or 0).

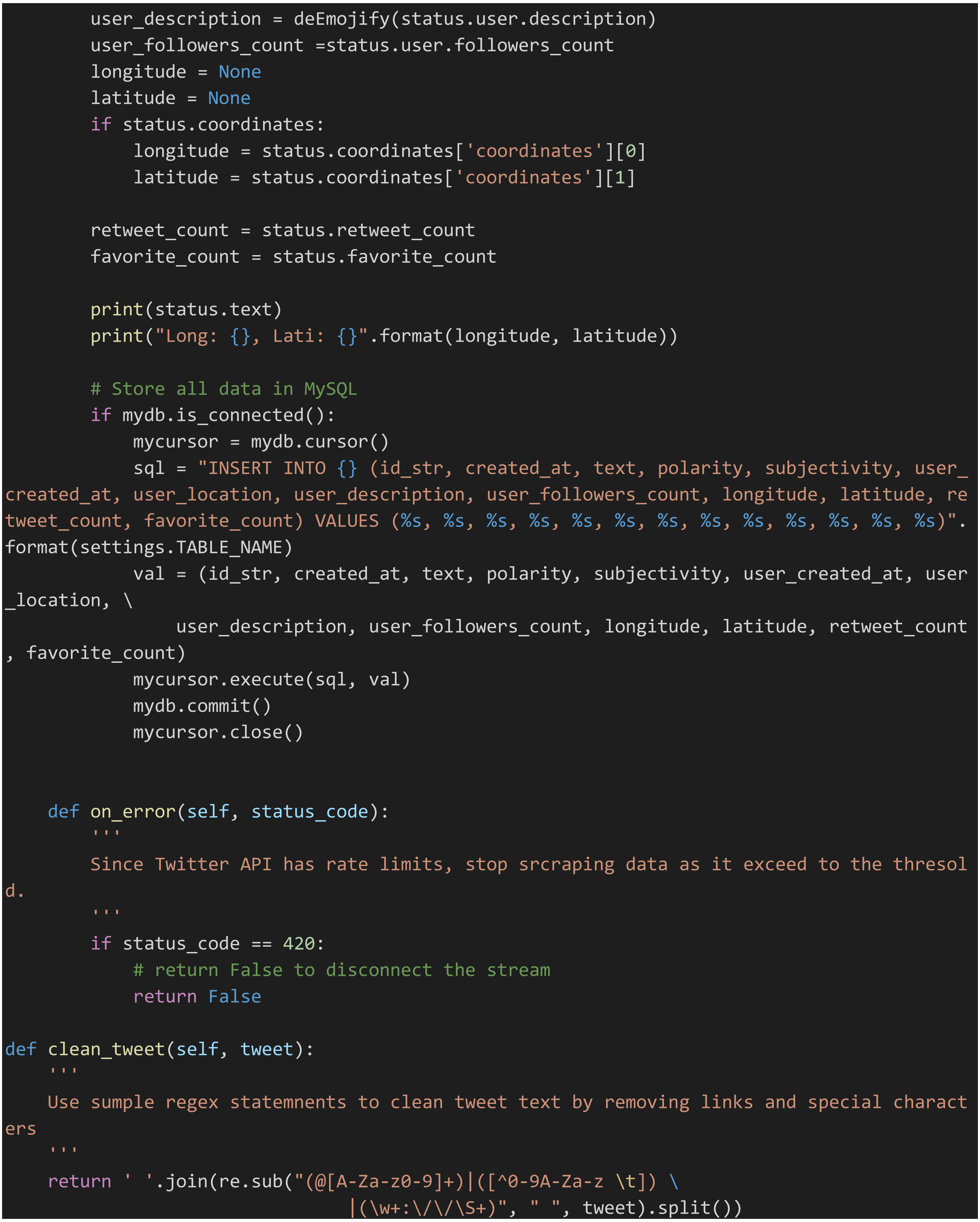


CHAPTER -4

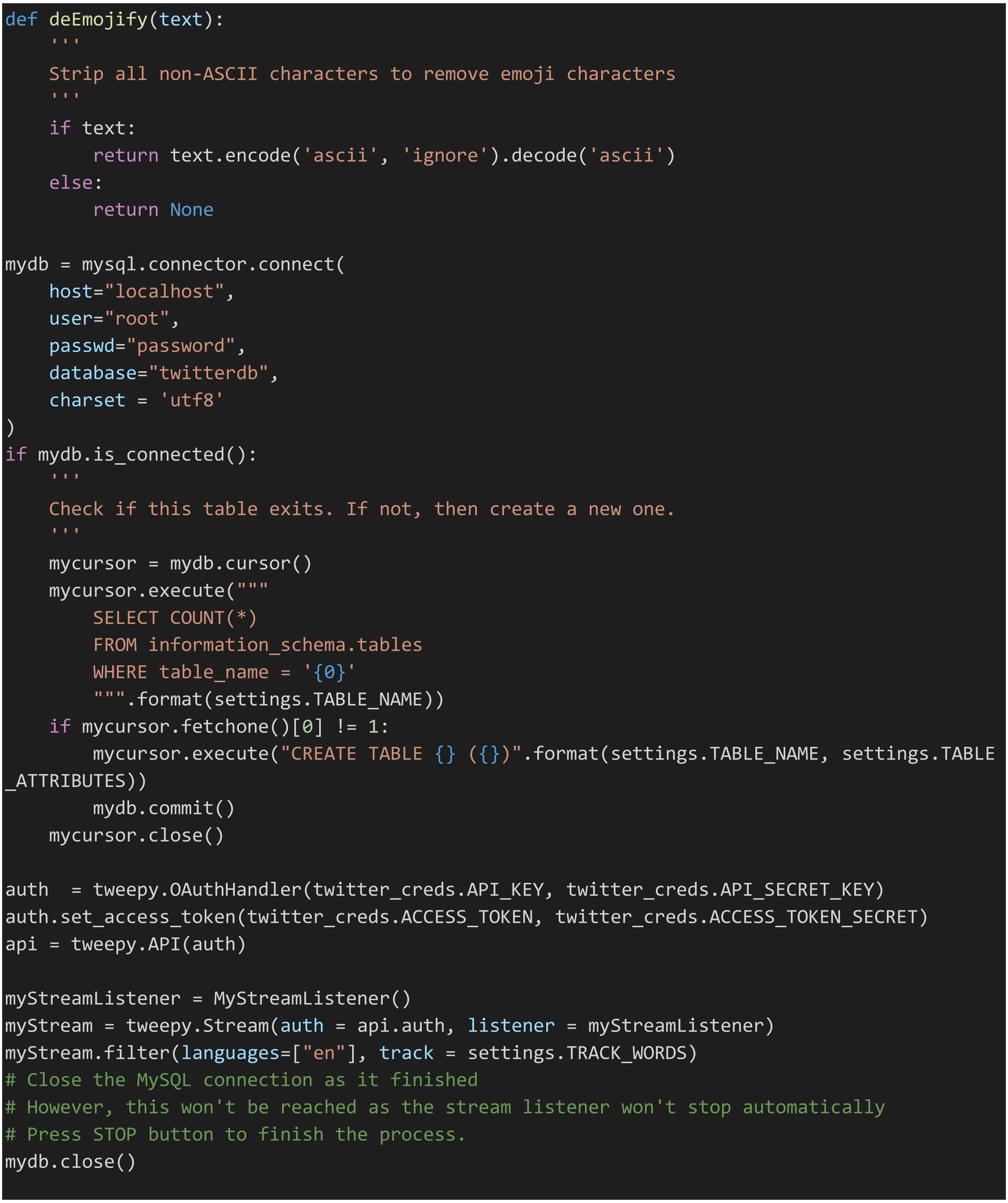
SYSTEM DESIGN CODE FOR SCRAPING AND ANALYSIS



```
# Extracting streaming data from Twitter, pre-processing, and loading into MySQL
import twitter_creds # Import api/access_token keys from credentials.py
import settings # Import related setting constants from settings.py
import re
import tweepy
import mysql.connector
import pandas as pd
from textblob import TextBlob
# Streaming With Tweepy

class MyStreamListener(tweepy.StreamListener):
    '''
    Tweets are known as “status updates”. So the Status class in tweepy has properties describing the tweet.

    '''

    def on_status(self, status):
        '''
        Extract info from tweets
        '''

        if status.retweeted:
            # Avoid retweeted info, and only original tweets will be received
            return True
        # Extract attributes from each tweet
        id_str = status.id_str
        created_at = status.created_at
        text = deEmojify(status.text)    # Pre-processing the text
        sentiment = TextBlob(text).sentiment
        polarity = sentiment.polarity
        subjectivity = sentiment.subjectivity

        user_created_at = status.user.created_at
        user_location = deEmojify(status.user.location)
```



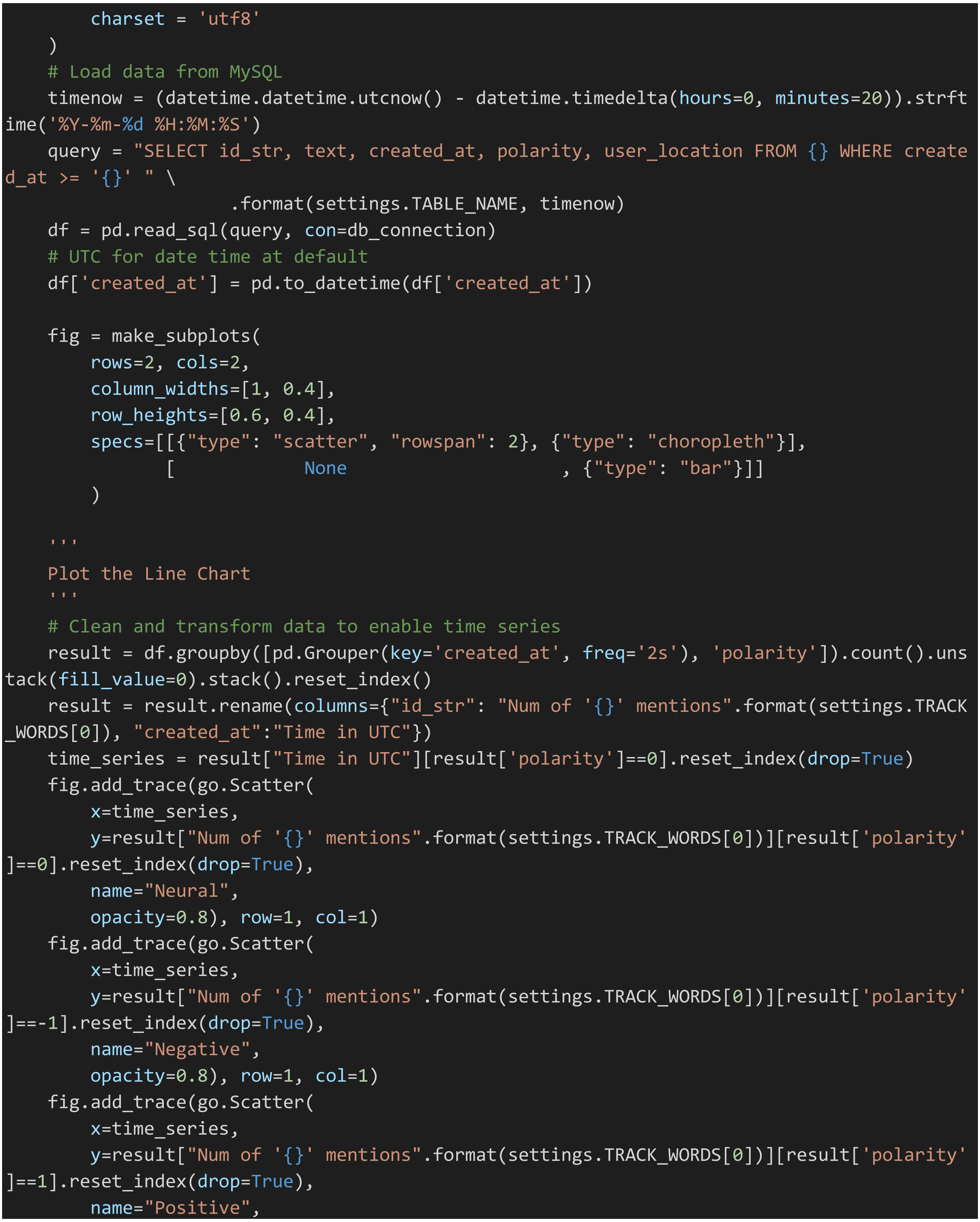
```
        user_description = deEmojify(status.user.description)
        user_followers_count =status.user.followers_count
        longitude = None
        latitude = None
        if status.coordinates:
            longitude = status.coordinates['coordinates'][0]
            latitude = status.coordinates['coordinates'][1]

        retweet_count = status.retweet_count
        favorite_count = status.favorite_count

        print(status.text)
        print("Long: {}, Lati: {}".format(longitude, latitude))

        # Store all data in MySQL
        if mydb.is_connected():
            mycursor = mydb.cursor()
            sql = "INSERT INTO {} (id_str, created_at, text, polarity, subjectivity, user_
created_at, user_location, user_description, user_followers_count, longitude, latitude, re
tweet_count, favorite_count) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)".
format(settings.TABLE_NAME)
            val = (id_str, created_at, text, polarity, subjectivity, user_created_at, user
_location, \
                user_description, user_followers_count, longitude, latitude, retweet_count
, favorite_count)
            mycursor.execute(sql, val)
            mydb.commit()
            mycursor.close()


    def on_error(self, status_code):
        '''
        Since Twitter API has rate limits, stop srcraping data as it exceed to the thresol
d.
        '''
        if status_code == 420:
            # return False to disconnect the stream
            return False

def clean_tweet(self, tweet):
    '''
    Use sumple regex statemnents to clean tweet text by removing links and special charact
ers
    '''
    return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t]) \
                                |(\w+:\/\/\S+)", " ", tweet).split())
```



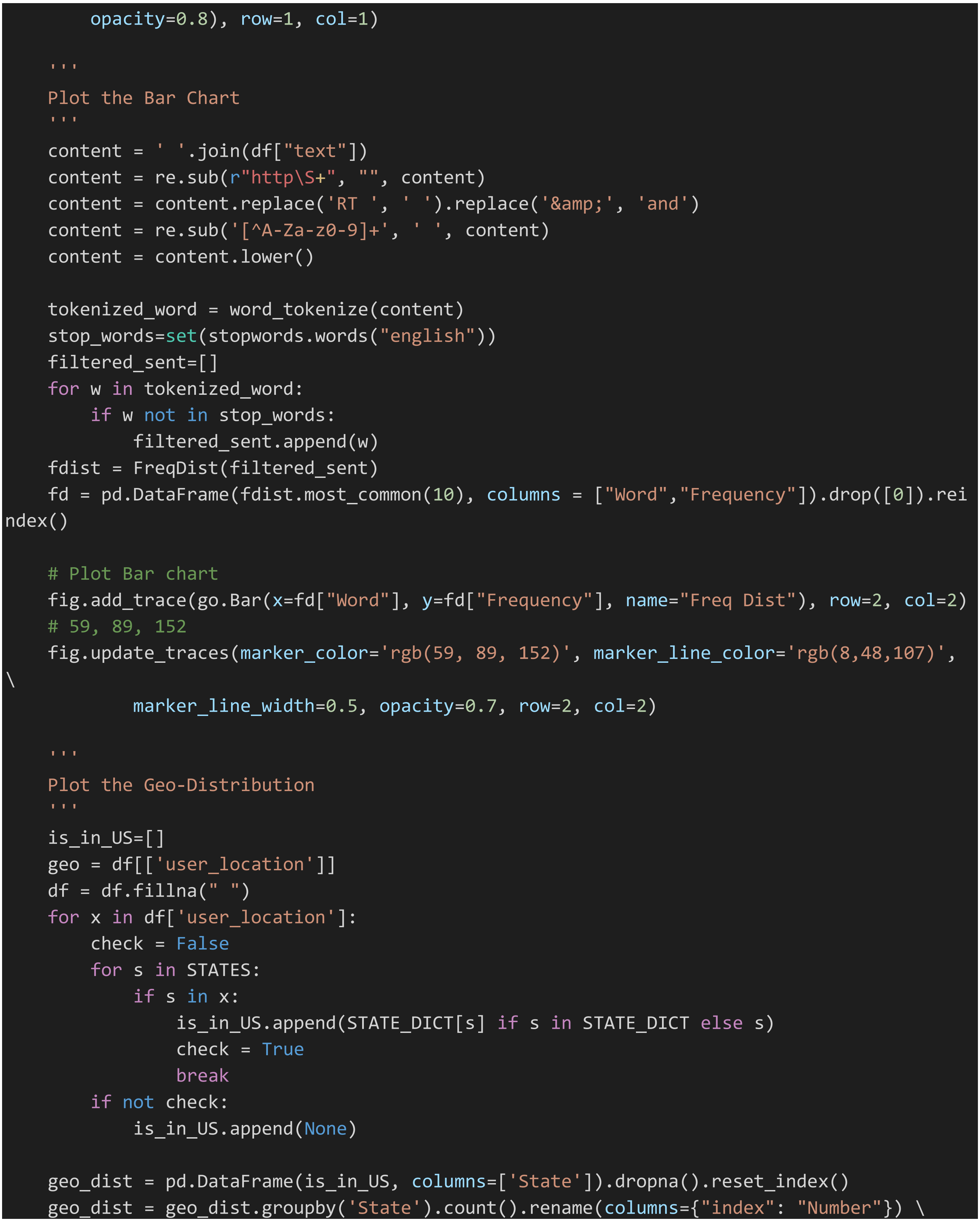
```
def deEmojify(text):
    '''
    Strip all non-ASCII characters to remove emoji characters
    '''
    if text:
        return text.encode('ascii', 'ignore').decode('ascii')
    else:
        return None

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="password",
    database="twitterdb",
    charset = 'utf8'
)
if mydb.is_connected():
    '''
    Check if this table exits. If not, then create a new one.
    '''
    mycursor = mydb.cursor()
    mycursor.execute("""
        SELECT COUNT(*)
        FROM information_schema.tables
        WHERE table_name = '{0}'
        """.format(settings.TABLE_NAME))
    if mycursor.fetchone()[0] != 1:
        mycursor.execute("CREATE TABLE {} ({})".format(settings.TABLE_NAME, settings.TABLE
_ATTRIBUTES))
        mydb.commit()
    mycursor.close()

auth  = tweepy.OAuthHandler(twitter_creds.API_KEY, twitter_creds.API_SECRET_KEY)
auth.set_access_token(twitter_creds.ACCESS_TOKEN, twitter_creds.ACCESS_TOKEN_SECRET)
api = tweepy.API(auth)

myStreamListener = MyStreamListener()
myStream = tweepy.Stream(auth = api.auth, listener = myStreamListener)
myStream.filter(languages=["en"], track = settings.TRACK_WORDS)
# Close the MySQL connection as it finished
# However, this won't be reached as the stream listener won't stop automatically
# Press STOP button to finish the process.
mydb.close()
```

SYSTEM DESIGN CODE FOR VISUALIZATION

```
import settings
import mysql.connector
import pandas as pd
import time
import itertools
import math
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
#%matplotlib inline
import plotly.express as px
import datetime
from IPython.display import clear_output
import plotly.offline as py
import plotly.graph_objs as go
from plotly.subplots import make_subplots
py.init_notebook_mode()
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

# Filter constants for states in US
STATES = ['']
STATE_DICT = dict(itertools.zip_longest(*[iter(STATES)] * 2, fillvalue=""))
INV_STATE_DICT = dict((v,k) for k,v in STATE_DICT.items())
'''
This complex plot shows the latest Twitter data within 20 mins and will automatically upda
te.
'''
while True:
    clear_output()
    db_connection = mysql.connector.connect(
        host="localhost",
        user="root",
        passwd="password",
        database="twitterdb",
```



```
        charset = 'utf8'
    )
    # Load data from MySQL
    timenow = (datetime.datetime.utcnow() - datetime.timedelta(hours=0, minutes=20)).strft
ime('%Y-%m-%d %H:%M:%S')
    query = "SELECT id_str, text, created_at, polarity, user_location FROM {} WHERE create
d_at >= '{}' " \
                     .format(settings.TABLE_NAME, timenow)
    df = pd.read_sql(query, con=db_connection)
    # UTC for date time at default
    df['created_at'] = pd.to_datetime(df['created_at'])

    fig = make_subplots(
        rows=2, cols=2,
        column_widths=[1, 0.4],
        row_heights=[0.6, 0.4],
        specs=[[{"type": "scatter", "rowspan": 2}, {"type": "choropleth"}],
               [            None                    , {"type": "bar"}]]
        )

    '''
    Plot the Line Chart
    '''
    # Clean and transform data to enable time series
    result = df.groupby([pd.Grouper(key='created_at', freq='2s'), 'polarity']).count().uns
tack(fill_value=0).stack().reset_index()
    result = result.rename(columns={"id_str": "Num of '{}' mentions".format(settings.TRACK
_WORDS[0]), "created_at":"Time in UTC"})
    time_series = result["Time in UTC"][result['polarity']==0].reset_index(drop=True)
    fig.add_trace(go.Scatter(
        x=time_series,
        y=result["Num of '{}' mentions".format(settings.TRACK_WORDS[0])][result['polarity'
]==0].reset_index(drop=True),
        name="Neural",
        opacity=0.8), row=1, col=1)
    fig.add_trace(go.Scatter(
        x=time_series,
        y=result["Num of '{}' mentions".format(settings.TRACK_WORDS[0])][result['polarity'
]==-1].reset_index(drop=True),
        name="Negative",
        opacity=0.8), row=1, col=1)
    fig.add_trace(go.Scatter(
        x=time_series,
        y=result["Num of '{}' mentions".format(settings.TRACK_WORDS[0])][result['polarity'
]==1].reset_index(drop=True),
        name="Positive",
```

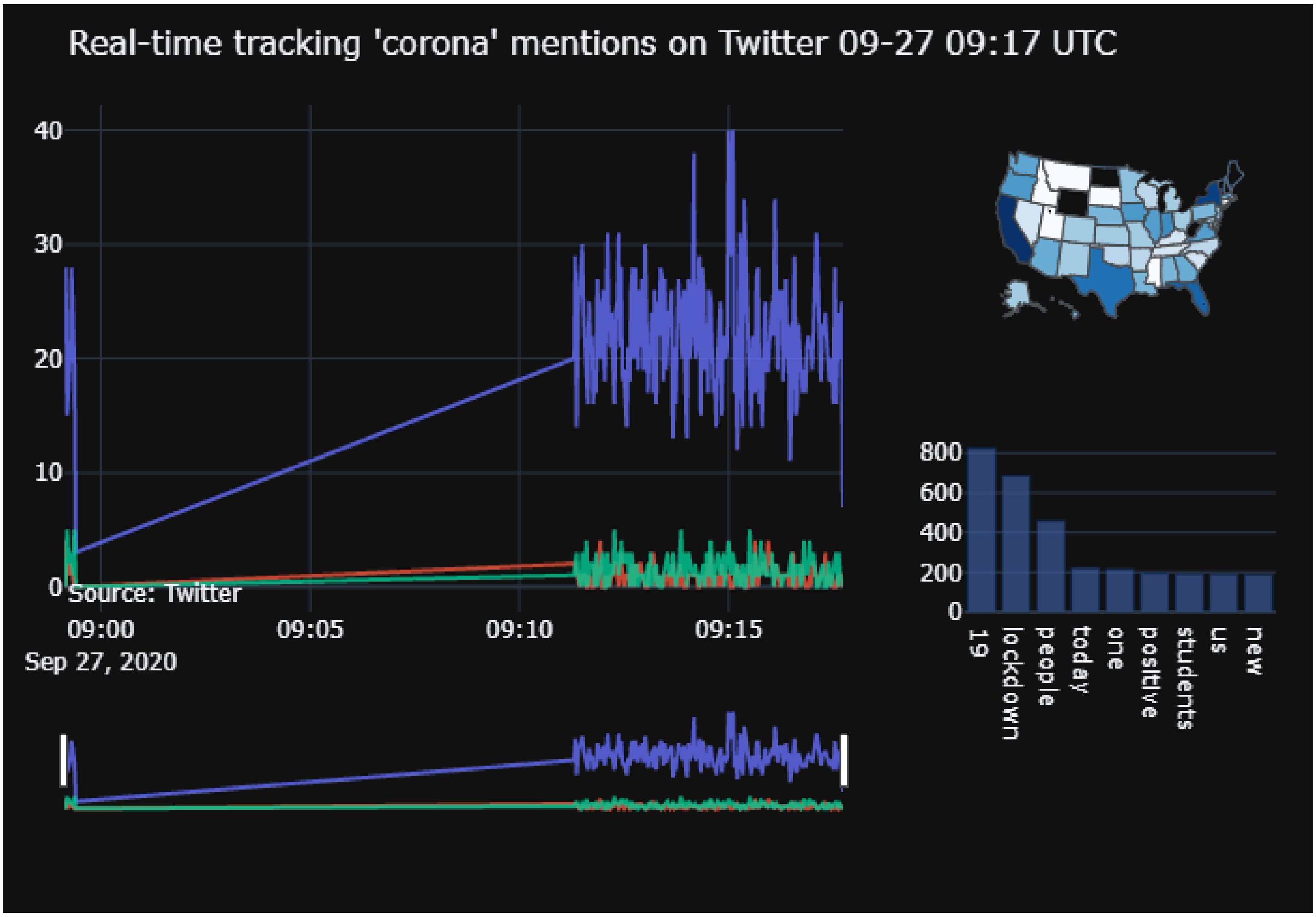


```
        opacity=0.8), row=1, col=1)

    '''
    Plot the Bar Chart
    '''
    content = ' '.join(df["text"])
    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()

    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]).rei
ndex()

    # Plot Bar chart
    fig.add_trace(go.Bar(x=fd["Word"], y=fd["Frequency"], name="Freq Dist"), row=2, col=2)
    # 59, 89, 152
    fig.update_traces(marker_color='rgb(59, 89, 152)', marker_line_color='rgb(8,48,107)',
\
            marker_line_width=0.5, opacity=0.7, row=2, col=2)

    '''
    Plot the Geo-Distribution
    '''
    is_in_US=[]
    geo = df[['user_location']]
    df = df.fillna(" ")
    for x in df['user_location']:
        check = False
        for s in STATES:
            if s in x:
                is_in_US.append(STATE_DICT[s] if s in STATE_DICT else s)
                check = True
                break
        if not check:
            is_in_US.append(None)

    geo_dist = pd.DataFrame(is_in_US, columns=['State']).dropna().reset_index()
    geo_dist = geo_dist.groupby('State').count().rename(columns={"index": "Number"}) \
            .sort_values(by=['Number'], ascending=False).reset_index()
    geo_dist["Log Num"] = geo_dist["Number"].apply(lambda x: math.log(x, 2))


    geo_dist['Full State Name'] = geo_dist['State'].apply(lambda x: INV_STATE_DICT[x])
    geo_dist['text'] = geo_dist['Full State Name'] + '<br>' + 'Num: ' + geo_dist['Number'].astype(str)
    fig.add_trace(go.Choropleth(
        locations=geo_dist['State'], # Spatial coordinates
        z = geo_dist['Log Num'].astype(float), # Data to be color-coded
        locationmode = 'USA-states', # set of locations match entries in `locations`
        colorscale = "Blues",
        text=geo_dist['text'], # hover text
        showscale=False,
        geo = 'geo'
        ),
        row=1, col=2)

    fig.update_layout(
        title_text= "Real-time tracking '{}' mentions on Twitter {} UTC".format(settings.TRACK_WORDS[0] ,datetime.datetime.utcnow().strftime('%m-%d %H:%M')),
        geo = dict(
            scope='usa',
        ),
        template="plotly_dark",
        margin=dict(r=20, t=50, b=50, l=20),
        annotations=[
            go.layout.Annotation(
                text="Source: Twitter",
                showarrow=False,
                xref="paper",
                yref="paper",
                x=0,
                y=0)
        ],
        showlegend=False,
        xaxis_rangeslider_visible=True
    )

    fig.show()

    time.sleep(60)
```

CHAPTER 5

RESULTS/CONCLUSION

Shown below is an instance captured from a real-time live dashboard based upon custom data set collected via Twitter Scraping.



CHAPTER-6

REFERENCES

- https://towardsdatascience.com/twitter-sentiment-analysis-based-on-news-topics-during-covid-19-c3d738005b55
- https://arxiv.org/pdf/2004.13932.pdf
- https://towardsdatascience.com/real-time-twitter-sentiment-analysis-for-brand-improvement-and-topic-tracking-chapter-1-3-e02f7652d8ff
- https://towardsdatascience.com/real-time-twitter-sentiment-analysis-for-brand-improvement-and-topic-tracking-chapter-2-3-1caf05346721