
PROJECT REPORT FOR

SBSPS_Challenge_2653

Sentiment Analysis Of COVID-19 Tweets - Visualization Dashboard

Team Name: TECHNOCRAT
Team Member(Leader): Snehal Mangesh Jadhav
Date of submission : 14th July 2020

---

1. INTRODUCTION

Sentiment analysis is the process of using natural language processing, text analysis, and statistics to analyze the sentiment of people.It is uesd mostly for the following purposes

Sentiment analysis for brand monitoring

One of the most well documented uses of sentiment analysis is to get a full 360 view of how your brand, product, or company is viewed by your customers and stakeholders. Widely available media, like product reviews and social, can reveal key insights about what your business is doing right or wrong. Companies can also use sentiment analysis to measure the impact of a new product, ad campaign, or consumer's response to recent company news on social media.

Sentiment analysis for customer service

Customer service agents often use sentiment or intent analysis to automatically sort incoming user email into "urgent" or "not urgent" buckets based on the sentiment of the email, proactively identifying frustrated users. The agent then directs their time toward resolving the users with the most urgent needs first. As customer service becomes more and more automated through machine learning, understanding the sentiment and intent of a given case becomes increasingly important.

Sentiment analysis for market research and analysis

Sentiment analysis is used in business intelligence to understand the subjective reasons why consumers are or are not responding to something (e.x. why are consumers buying a product? What do they think of the user experience? Did customer service support meet their expectations?). Sentiment analysis can also be used in the areas of political science, sociology, and psychology to analyze trends, ideological bias, opinions, gauge reactions, etc.

1.1 Overview

The Sentiment Analysis of twitter tweets project deals with the machine learning algorithm for analysis of tweets and dash framework for visualization dashboard.There are thousands of tweets tweeted daily on various topics and happenings from which we can understand the attitude of the people on the particular topic.In this project , I tried to pull the live tweets from the twitter using twitter API keys which I got from my twitter developer account.There are many in-built python libraries that does the half work for integrating the twitter tweets with python code for preprocessing and analysis. The result got from the analysis is visualized using dash framework. Dash framework is used for building the web app that can easily be deployed using any Platform like IBM Cloud foundry available on IBM Cloud.

1.2 Purpose

The major purpose of building this project is to identify the stress or anxiety level caused due to the pandemic that has affected the people from length and breadth of all the countries in the world.So that , proper decisions would be taken by the authorities to make lives of people better and normal.

2. LITERATURE SURVEY

2.1 Existing Problem

Due to the pandemic of COVID-19, there has been increased levels in stress and anxiety among the people.To understand the seniments of the people is necesssary for the authorities to know the opinions of people for implementing any new regulations or restrictions in country.

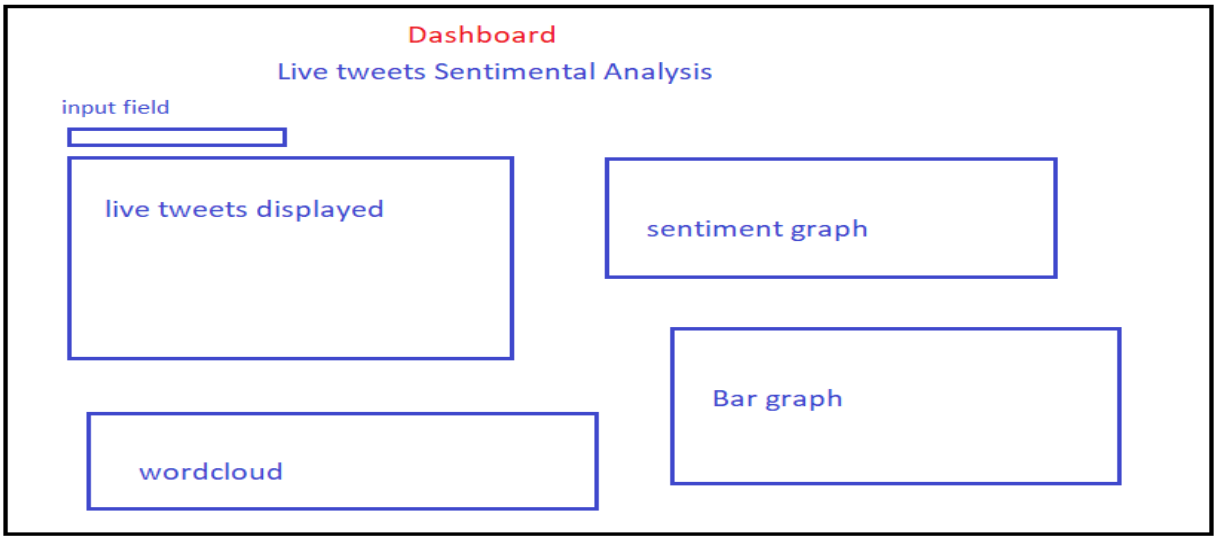
2.2 Proposed solution

To perform Sentiment Analysis on tweets of twitter, I am going to build the machine learning model which will use python 3 as the programming language.The IBM platform Watson Studio will be used for developing the project.Along with this dataset for analysis will be taken from twitter API.Using python packages and machine learning algorithm classification and prediction of sentiments of people during lockdown will be done.

3. THEORITICAL ANALYSIS

3.1 Block Diagram

The Block diagram for Project



3.2 Hardware and Software Designing

Required Hardware:

Proccessor:Intel Core i5(8th Gen){used}

RAM:8 GB

Storage:more than 256 MB

Software:

OS : Windows ,Linux

IBM Cloud Lite Services

Python 3

IBM Command Line Interface

GIT

Jupyter Notebook or any Python IDLE

4. EXPERIMENTAL INVESTIGATIONS

STEP 1: Choose a Project Idea.

Project Idea was to build Sentiment Analysis using nltk package in python.Creation of dashboard using Dash framework which is built on python.

STEP 2: Conduct Background Research.

Studied some of the research paper on sentiment Analysis and watched youtube videos on it.Technical Help taken from IBM mentors and Bootcamps arranged by IBM.

STEP 3: Compose a Hypothesis.

It was pressumed about the sentiments of people to be negative on the pandemic and its effects on the lifestyle of people.

STEP 4: Design Your **Experiment**.

A template was designed for the dashboard and its components to be included.

STEP 5: Collect Data.

As it was a live streaming of data, collection of data was not required.

STEP 6: Analyze Your Data and Draw a Conclusion.

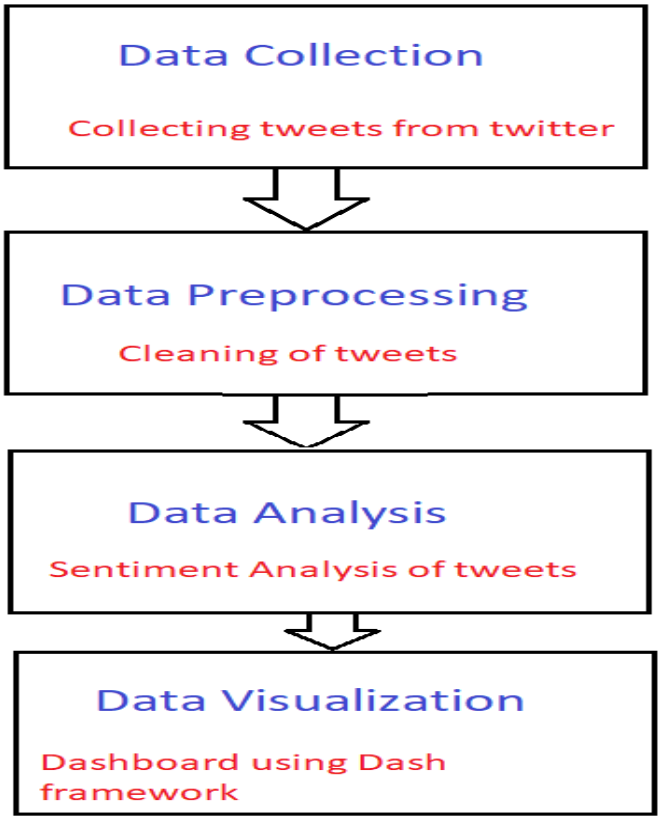
After completion of the project , its output came as the hypothesis was made.The sentiment graph went towards the negative side when did for COVID-19 and Lockdown.

STEP 7: Communicate Your Results.

The results are variable due the live tweets

5. FLOWCHART

The flowcart for project is as follows:



6. RESULT

The result of the Sentiment Analysis is people are not happy with the lockdown extension and on the uncontrollable spread of COVID-19 virus.

7. ADVANTAGES & DISADVANTAGES

Advantages:

1. Using sentiment Analysis on social media helps us to find the emotions and mind-set of people on certain topic , very quickly and easily.
2. Predictions can be done accurately and strategies can be built through it.
3. Whole insights are obtained on the particular happening or topic.
4. Decision making becomes easier due to predictions done through the results obtained.

Disadvantages:

1. Sentiment analysis cannot be accurately done due to the limited scope of data.
2. Machine Learning models cannot understand sentiments from certain words.Example: It cannot understand sarcasm.

8. APPLICATIONS

1.It can be used for determing the mental health of people
2.It can be used for planning and implementing policies for authorities.

9. CONCLUSION

Nowadays, sentiment analysis is an important topic in machine learning. We are still far to detect the sentiments of a corpus of texts very accurately because of the complexity in the English language and even more if we consider other languages such as Japanese or Mandarin. In this project I tried to show the basic way of classifying tweets into positive or negative category using Natural Language Proccessing as baseline . We could further improve our classifier by trying to extract more features from the tweets, trying different kind of machine learning algorithms.

10. FUTURE SCOPE

The is lot of scope for sentiment analysis as it is a very new application of machine learning.Understanding the correct meaning of the words and making clusters of emotions has to be more accurate.Tweets have lot of features that can be studied for more better results.

Code:

Data_Extraction.py

```
from tweepy import Stream
from tweepy import OAuthHandler
import json
import sqlite3
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from unidecode import unidecode
import time
from datetime import datetime
import re
import nltk
import string
```

```
from tweepy.streaming import StreamListener
import numpy as np
from nltk.stem.porter import *
from nltk.tokenize import word_tokenize
from string import punctuation
from nltk.corpus import stopwords

analyzer = SentimentIntensityAnalyzer()

account, and add your key details here
ckey="XXXX"
csecret="XXXXX"
atoken="XXXX"
asecret="XXXX"

stemmer = PorterStemmer()

class PreProcessTweets:
    def __init__(self):
                self._stopwords  =  set(stopwords.words('english')  +
list(punctuation) + ['AT_USER','URL', 'RT', "n't", "..."])
    def processTweets(self, tweet):
        tweet = tweet.lower() # convert text to lower-case
            tweet  =  re.sub('((www\.[^\s]+)|(https?://[^\s]+))',  'URL',
tweet) # remove URLs
        tweet = re.sub('@[^\s]+', 'AT_USER', tweet) # remove usernames
          tweet = re.sub(r'#([^\s]+)', r'\1', tweet) # remove the # in
#hashtag
            tweet  =  word_tokenize(tweet)  # remove repeated characters
(hellooooooooo into hello)
        return [word for word in tweet if (word not in self._stopwords)
& (len(word) > 2)]

conn = sqlite3.connect('twitter.db')
c = conn.cursor()

def create_table():
```

```
    try:
            c.execute("CREATE TABLE IF NOT EXISTS sentiment(unix REAL,
rdtime DATETIME, tweet TEXT, cleanedtweet TEXT, \
                  sentiment REAL)")
        c.execute("CREATE INDEX fast_unix ON sentiment(unix)")
        c.execute("CREATE INDEX fast_rdtime ON sentiment(rdtime)")
        c.execute("CREATE INDEX fast_tweet ON sentiment(tweet)")
                        c.execute("CREATE   INDEX   fast_sentiment   ON
sentiment(sentiment)")
        conn.commit()
    except Exception as e:
        print(str(e))
create_table()

class listener(StreamListener):

    def on_data(self, data):
        try:
            texttweet = ""
            data = json.loads(data)
            tweet = unidecode(data['text'])
            time_ms = data['timestamp_ms']
                                                    rd_time_ms     =
datetime.utcfromtimestamp(int(time_ms)/1000).strftime('%Y-%m-%d
%H:%M:%S')
            vs = analyzer.polarity_scores(tweet)
            tweetProcessor = PreProcessTweets()
            cleanedtweet = tweetProcessor.processTweets(tweet)
            texttweet = " ".join(cleanedtweet)
            expected_keys = ['user', 'location']
            sentiment = vs['compound']
            print(time_ms, rd_time_ms, tweet, texttweet, sentiment)
            #print(texttweet)
                c.execute("INSERT INTO sentiment (unix, rdtime, tweet,
cleanedtweet, sentiment) VALUES (?, ?, ?, ?, ?)",
                  (time_ms, rd_time_ms, tweet, texttweet, sentiment))
            conn.commit()
```

```
        except KeyError as e:
            print("Error: ", str(e))
        return(True)

    def on_error(self, status):
        print(status)
while True:
    try:
        auth = OAuthHandler(ckey, csecret)
        auth.set_access_token(atoken, asecret)
        twitterStream = Stream(auth, listener())
          # Modify the word for which we are searching tweets, in this
case used "covid19".

        twitterStream.filter(track=["covid19"])
    except Exception as e:
        print(str(e))
        time.sleep(5)
```

Code :
Dashboard.py

```
import dash
from dash.dependencies import Output, Input
import dash_core_components as dcc
import dash_html_components as html
import plotly
import dash_table
import random
import plotly.graph_objs as go
from collections import deque
import sqlite3
import pandas as pd
import time
import datetime
from collections import Counter
```

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator


app = dash.Dash(__name__)

columns = [
    {"id": 0, "name": "unix"},
    {"id": 1, "name": "dateTime"},
    {"id": 2, "name": "tweet"},
    {"id": 3, "name": "cleanedtweet"},
    {"id": 4, "name": "sentiment"},
    {"id": 5, "name": "sentiment_smoothed"}
]

colors = {
    'background': '#111111',
    'text': '#7FDBFF'
}

app.layout = html.Div(style={'backgroundColor': colors['background']},
children=[
        html.Div([html.H1(children='Live Twitter Sentiment Analytics',
                                style={'textAlign': 'center', 'color':
colors['text']})]),
        html.Div([html.H2('Dashboard',
                                style={'textAlign': 'center', 'color':
colors['text']})]),
           html.Div([html.H3("Enter the term you like to do sentiment
analysis",
                style={'textAlign': 'left', 'color': colors['text']}),
                    dcc.Input(id='sentiment_term', value='Coronavirus',
type='text')]),
        html.Div([
            html.Div([dcc.Graph(id='live-graph', animate=False),
                      dcc.Interval(id='graph-update', interval=1*1000,
n_intervals=0),
                html.Br()
```

```
                ], style={'width': '49%', 'float': 'right', 'display':
'inline-block'}),
            html.Div([
                dash_table.DataTable(
                    style_data={'whiteSpace': 'normal'},
                                        css=[{'selector':  '.dash-cell
div.dash-cell-value',
                                 'rule': 'display: inline; white-space:
inherit; overflow: inherit; text-overflow: inherit;'}],
                    id='tweet-table',
                    data=[],
                    columns=[],
                        style_table={'maxHeight': '450px', 'overflowY':
'scroll','border': 'thin lightgrey solid',
                                     'overflowX': 'scroll'},
                         style_header={'backgroundColor': 'rgb(30, 30,
30)'},
                     style_cell={'minWidth': '150px', 'backgroundColor':
'rgb(50, 50, 50)', 'color': 'white'}),
                        dcc.Interval(id='table-update', interval=1*1000,
n_intervals=0),
                    html.Br()
                  ],style={'width': '49%', 'float': 'left', 'display':
'inline-block'})
        ]),
        html.Div([
            html.Div([
                dcc.Graph(id='bar-graph', animate=False),
                         dcc.Interval(id='bar-update',  interval=1*1000,
n_intervals=0)
                ], style={'width': '44%', 'float': 'right', 'display':
'inline-block'})
            ]),

html.Div(style={'backgroundColor': colors['background']},children=[
html.H3("WordCloud",style='textAlign':'center','color':colors['text']},
                html.Button(['Refresh WordCloud'], id='refbutton'),
```

```
                                        html.P(id='placeholder'),       #
app.get_asset_url('word1.png')
                            html.Div([html.Img(id='image',  height="369"  ,
width="680")
                    ], style={'width': '44%', 'float': 'top', 'display':
'inline-block'})
            ])
    ])

@app.callback(Output('bar-graph', 'figure'),
                  [Input('sentiment_term', 'value'), Input('bar-update',
'n_intervals')])
def generate_bar(sentiment_term, n_clicks):
    try:
        conn = sqlite3.connect('twitter.db')
        c = conn.cursor()
          df = pd.read_sql("SELECT * FROM sentiment WHERE tweet LIKE ?
ORDER BY unix DESC LIMIT 200", conn ,
                                  params=('%' + sentiment_term + '%',))
        df.sort_values('unix', inplace=True)
        df.dropna(inplace=True)
        stopwords = set(STOPWORDS)
        df=df[-100:]
        text = " ".join(review for review in df.cleanedtweet)
        split_it = text.split()
        counter = Counter(split_it)
        most_occur = counter.most_common(15)
        df1=pd.DataFrame(most_occur)
        X = df1[0]
        Y = df1[1]
        data = plotly.graph_objs.Bar(x=X, y=Y, name='Bar')

return
{'data':[data], 'layout' : go.Layout(plot_bgcolor=colors['background'],

paper_bgcolor=colors['background'],
```

```
font={'color': colors['text']},

title='Bar Chart - Most Frequent Term')}

    except Exception as e:
        with open('errors.txt','a') as f:
            f.write(str(e))
            f.write('\n')



@app.callback(Output(component_id='image', component_property='src'),

[Input(component_id='sentiment_term', component_property='value'),
             Input("refbutton", "n_clicks")])
def generate_wordcloud(sentiment_term, n_clicks):
    try:
        src = 'assets/word1.png'
if n_clicks is not None:
            conn = sqlite3.connect('twitter.db')
            c = conn.cursor()
             df = pd.read_sql("SELECT * FROM sentiment WHERE tweet LIKE
? ORDER BY unix DESC LIMIT 200", conn ,
                                 params=('%' + sentiment_term + '%',))
            df.sort_values('unix', inplace=True)
            df.dropna(inplace=True)
            stopwords = set(STOPWORDS)
            df=df[-100:]
            text = " ".join(review for review in df.cleanedtweet)
                           wordcloud  =  WordCloud(stopwords=stopwords,
background_color="black").generate_from_text(text)
                 ts  =  str(datetime.datetime.now().time()).replace(":",
"_").replace(".","_")
            wordcloud.to_file("assets/word_"+ts+".png")
            src = "assets/word_"+ts+".png"
        return src
```

```
    except Exception as e:
        with open('errors.txt','a') as f:
            f.write(str(e))
            f.write('\n')

@app.callback(Output('live-graph', 'figure'),


[Input(component_id='sentiment_term', component_property='value'),

Input(component_id='graph-update', component_property='n_intervals')])
def update_graph_scatter(sentiment_term, n):
    try:
        conn = sqlite3.connect('twitter.db')
        c = conn.cursor()
          df = pd.read_sql("SELECT * FROM sentiment WHERE tweet LIKE ?
ORDER BY unix DESC LIMIT 200", conn ,

params=('%' + sentiment_term + '%',))
        df.sort_values('unix', inplace=True)

df['sentiment_smoothed']                                                =
df['sentiment'].rolling(int(len(df)/10)).mean()
        df.dropna(inplace=True)
        X = df.rdtime.values[-100:]
        Y = df.sentiment_smoothed.values[-100:]

data   =   plotly.graph_objs.Scatter(x=X,   y=Y,   name='Scatter',   mode=
'lines+markers')

return
{'data': [data],'layout' : go.Layout(xaxis=dict(range=[min(X),max(X)]),

yaxis=dict(range=[min(Y),max(Y)]),

plot_bgcolor=colors['background'],
```

```
paper_bgcolor=colors['background'],


font={'color': colors['text']},

title='Term: {}'.format(sentiment_term))}
    except Exception as e:
        with open('errors.txt','a') as f:
            f.write(str(e))
            f.write('\n')
@app.callback([Output('tweet-table', 'data'),
Output('tweet-table', 'columns')],

[Input(component_id='sentiment_term', component_property='value'),

Input(component_id='table-update', component_property='n_intervals')])

def update_graph_bar(sentiment_term, n):
    try:
        conn = sqlite3.connect('twitter.db')
        c = conn.cursor()
          df = pd.read_sql("SELECT * FROM sentiment WHERE tweet LIKE ?
ORDER BY unix DESC LIMIT 200", conn ,

params=('%' + sentiment_term + '%',))
        df.sort_values('unix', inplace=True)
        df.dropna(inplace=True)
        df=df[-100:]
        return df.values[0:15], columns[1:3]
    except Exception as e:
        with open('errors.txt','a') as f:
            f.write(str(e))
            f.write('\n')
if __name__ == '__main__':
    app.run_server(debug=False)
-----------------------------------------------------------------------
```