MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

BHOPAL (M.P.)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**MAJOR PROJECT**

**ON**

# TWITTER SENTIMENTAL ANALYSIS USING LIVE STREAM DATA

SUBMITTED IN FULFILLMENT FOR THE DEGREE OF BACHELOR OF TECHNOLOGY

SUBMITTED BY:

C. M. DHEERAJ VERMA (141112230)

T. VENKATA SIVA KUMAR (141112245)

A. KRISHNA CHANDAN (141112003)

SATENDRA PRATAP (141112306)

UNDER THE GUIDANCE OF:

**DR. MANASI GYANCHANDHANI**

SESSION 2017-2018

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

BHOPAL (M.P)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## CERTIFICATE

This is to certify that **C. M. DHEERAJ VARMA, T. VENKATA SIVA KUMAR, A. KRISHNA CHANDAN** and **SATENDRA PRATAP** students of B. Tech 4th Year (Computer Science & Engineering), have successfully completed their project "**TWITTER SENTIMENTAL ANALYSIS USING LIVE STREAM DATA**" in fulfilment of their major project in Computer Science & Engineering.

**DR.NAMITA TIWARI** **DR. MANASI GYANCHANDHANI**

(Project Coordinator) (Project Guide)

## MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

## BHOPAL (M.P).

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## DECLARATION

We, hereby, declare that the following report which is being presented in the Major Project Documentation entitled "TWITTER SENTIMENTAL ANALYSIS USING LIVE STREAM DATA" is the fulfilment of the requirements of the fourth year (eight semester) Major Project in the field of Computer Science And Engineering. It is an authentic documentation of our own original work carried out under the able guidance of DR. Manasi Gyanchandhani and the dedicated co-ordination of Dr. Namita Tiwari. The work has been carried out entirely at Maulana Azad National Institute of Technology, Bhopal. The following project and its report, in part or whole, has not been presented or submitted by us for any purpose in any other institute or organization. We, hereby, declare that the facts mentioned above are true to the best of our knowledge. In case of any unlikely discrepancy that may possibly occur, we will be the ones to take responsibility.

C. M. DHEERAJ VARMA (141112230)

T. VENKATA SIVA KUMAR (141112245)

A. KRISHNA CHANDAN (141112003)

SATENDRA PRATAP (141112306)

## ACKNOWLEDGEMENT

With due respect, we express our deep sense of gratitude to our respected guide Dr. Manasi Gyanchandhani, for her valuable help and guidance. We are thankful for the encouragement that she has given us in completing this project successfully. Her rigorous evaluation and constructive criticism was of great assistance.

It is imperative for us to mention the fact that this minor project could not have been accomplished without the periodic suggestions and advise of our project co-ordinator Dr. Namita Tiwari.

We are also grateful to our respected director Dr. N. S. Chaudhary for permitting us to utilize all the necessary facilities of the college.

Needless to mention is the additional help and support extended by our respected HOD, Dr. Meenu Chawla, in allowing us to use the departmental laboratories and other services.

We are also thankful to all the other faculty, staff members and laboratory attendants of our department for their kind co-operation and help.
Last but certainly not the least; we would like to express our deep appreciation towards our family members and batch mates for providing the much needed support and encouragement.

TABLE OF CONTENTS:

LIST OF TABLES & FIGURES:

## Abstract:

The rise of social media has generated tremendous interest among Internet users today. Data from these social networking sites can be used for a number of purposes, like prediction, marketing or sentiment analysis. Twitter is a widely used social media site for posting comments through short statuses. The millions of tweets received every year could be subjected to sentiment analysis. This project addresses the use of sentiment analysis that is classifying text according to the sentiments

expressed in them: positive, negative or neutral. We explain the process of storing, preparing and analysing twitter streaming data, then we examine the methods and tools available in python programming language to visualize the analysed data. We believe that using social networks and microblogs for efficient analysis of massive real-time data about the product, services, and global events i

s crucial for better decisions. Twitter's popularity as an information source has led to the development of applications and research in various domains. Humanitarian Assistance and Disaster Relief is one domain where information from Twitter is used to provide situational awareness to a crisis situation. Managerial decisions, stock prediction, improving traffic prediction are some example of the use cases getting favourites among researchers.

# 1. Introduction:

## 1.1 Why Text Analysis?

Everyday enormous amount of data is created from social networks, blogs and other media and diffused into the World Wide Web. This huge data contains very crucial opinion related information that can be used to benefit business and other aspects of commercial and scientific industries. Manual tracking and extraction of this useful information is not possible, thus, text analysis is required from texts over particular subject, are or product online. It is an application of natural language processing, computational linguistics and text analytics to identify subjective information from source data. It clubs the sentiments into categories like “positive” or “negative”. Thus, it determines the general attitude of the speaker or a writer with respect to the topic in context.

## 1.2 Why Natural Language Processing?

Natural Language Processing (NLP) is the technology dealing with our most ubiquitous product: human language, as it appears in emails, web pages, tweets, product descriptions, newspapers stories, social media, and scientific articles, in thousands of languages and varieties. In the past decade, successful natural language processing to applications have become part of our everyday experience, from spelling and grammar correction in word processors to machine translation on the web, from email spam detection to automatic question answering, from detecting people’s opinion about products or services to extracting appointments from your email. The greatest challenge of the text analysis is to design application-specific algorithms and technique that can analyse the human language linguistics accurately.

# 2. Theoretical aspect

## 2.1 What is a classifier?

A classifier is a system that performs a mapping from a feature space X to a set of labels Y. Basically what a classifier does is assign a pre-defined class label to sample. For example, if you are building a spam classifier then the feature space contains a representation of an email and the label is either “SPAM” or “NON-SPAM”.

This should not be confused with clustering where the algorithm autonomously partition the data into clusters in a way so that the data in each cluster is grouped in feature space. Classification is a supervised method where you decide the classes while clustering is an unsupervised method where the algorithm groups the data autonomously.

- **SVM**

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (*supervised learning*), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

Suppose you are given plot of two label classes on graph as shown in image *(A)*. Can you decide a separating line for the classes?

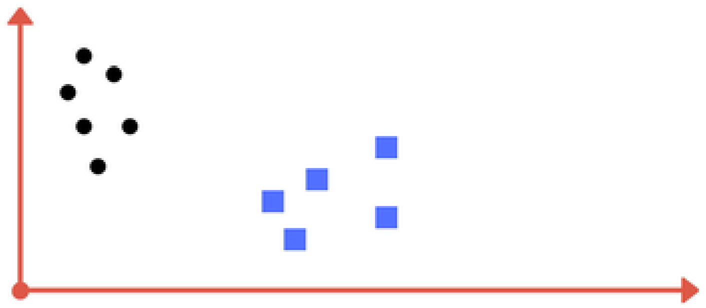


Figure 2.1: graph consisting of two classes

You might have come up with something similar to following image (*image B*). It fairly separates the two classes. Any point that is left of line falls into black circle class and on right falls into blue square class. *Separation of classes. That's what SVM does***.** It finds out a line/ hyper-plane (in multidimensional space that separate outs classes). Shortly, we shall discuss why I wrote multidimensional space.

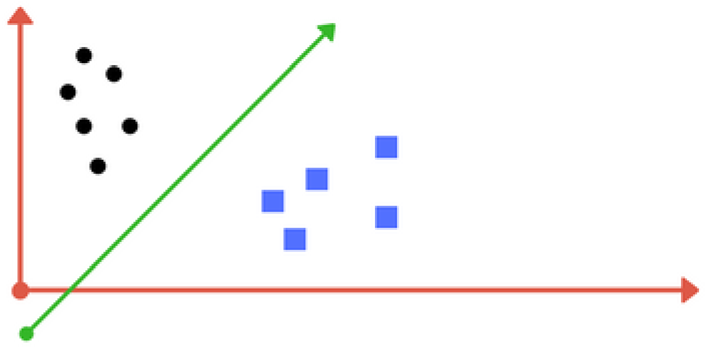


Figure 2.2: graph consisting of two classes

But what if the data is like this?

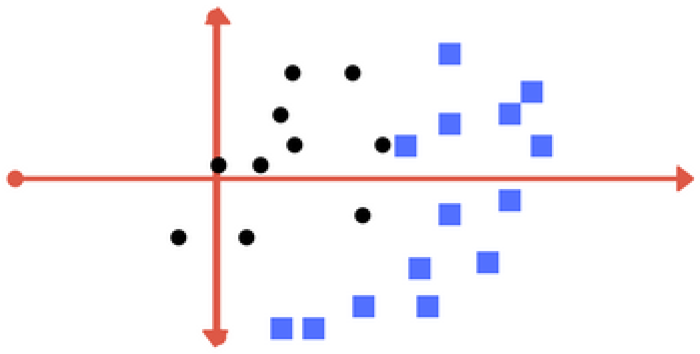


Figure 2.3: graph consisting of two classes

The solution for this would be something like this

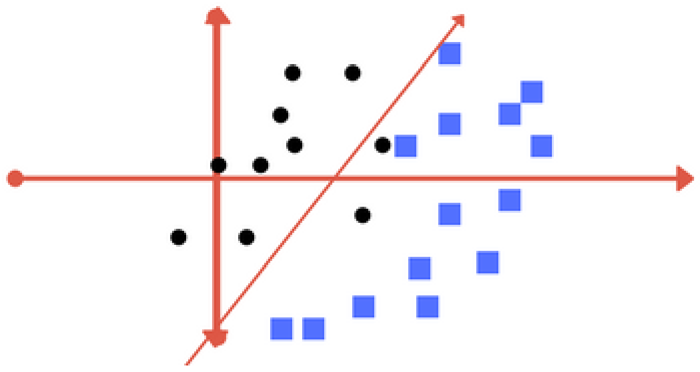


Figure 2.4: graph consisting of two classes

This is done by tuning parameters: Kernel, Regularization, Gamma and Margin.

- **Kernel**

The learning of the hyper plane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role.

For **linear kernel** the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

f(x) = B(0) + sum(ai * (x,xi))

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B0 and ai (for each input) must be estimated from the training data by the learning algorithm.

The **polynomial kernel** can be written as *K(x,xi) = 1 + sum(x * xi)^d* and **exponential** as *K(x,xi) = exp(-gamma * sum((x—xi²)).*
*Polynomial and exponential kernels calculates separation line in higher dimension. This is called* ***kernel trick***

- ## Regularization

The Regularization parameter (often termed as C parameter in python's sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example.

For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.The images below (same as image 1 and image 2 in section 2) are example of two different regularization parameter. Left one has some misclassification due to lower regularization value. Higher value leads to results like right one.

- ## Gamma

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Whereas high gamma means the points close to plausible line are considered in calculation.

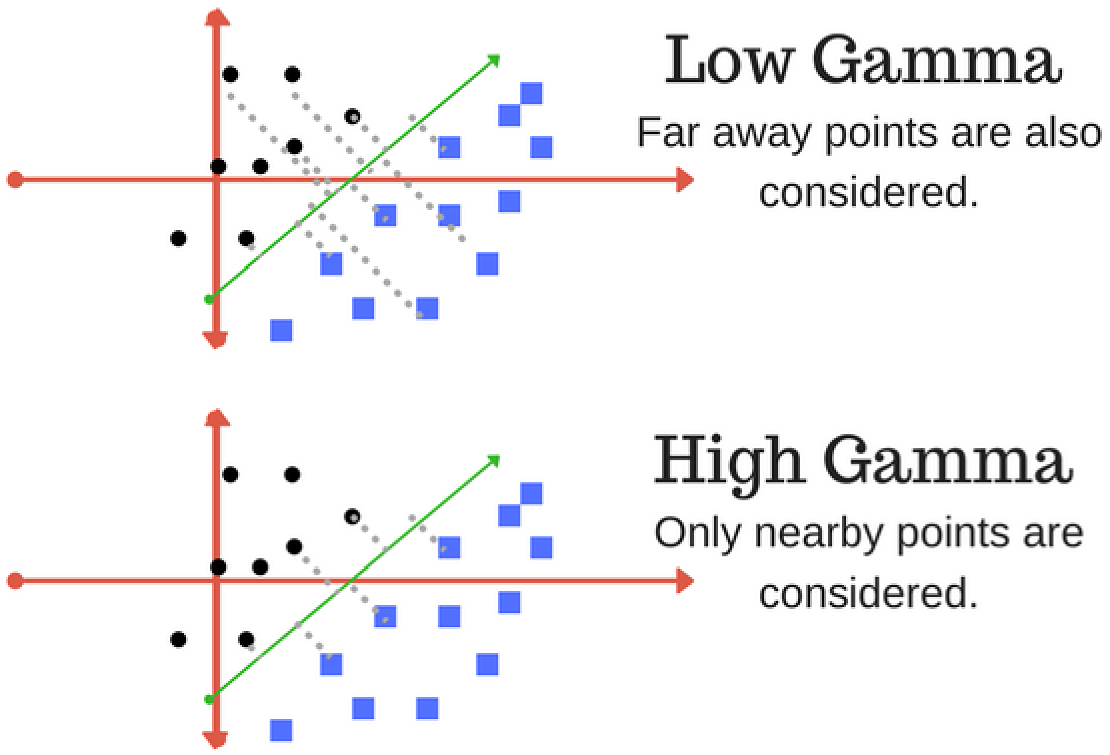

Low Gamma
Far away points are also considered.
High Gamma
Only nearby points are considered.

Figure 2.5: High and Low GAMA

- **Margin**

And finally last but very important characteristic of SVM classifier. SVM to core tries to achieve a good margin.

***A margin is a separation of line to the closest class points.***

A good margin is one where this separation is larger for both the classes. Images below gives to visual example of good and bad margin. A good margin allows the points to be in their respective classes without crossing to other class.

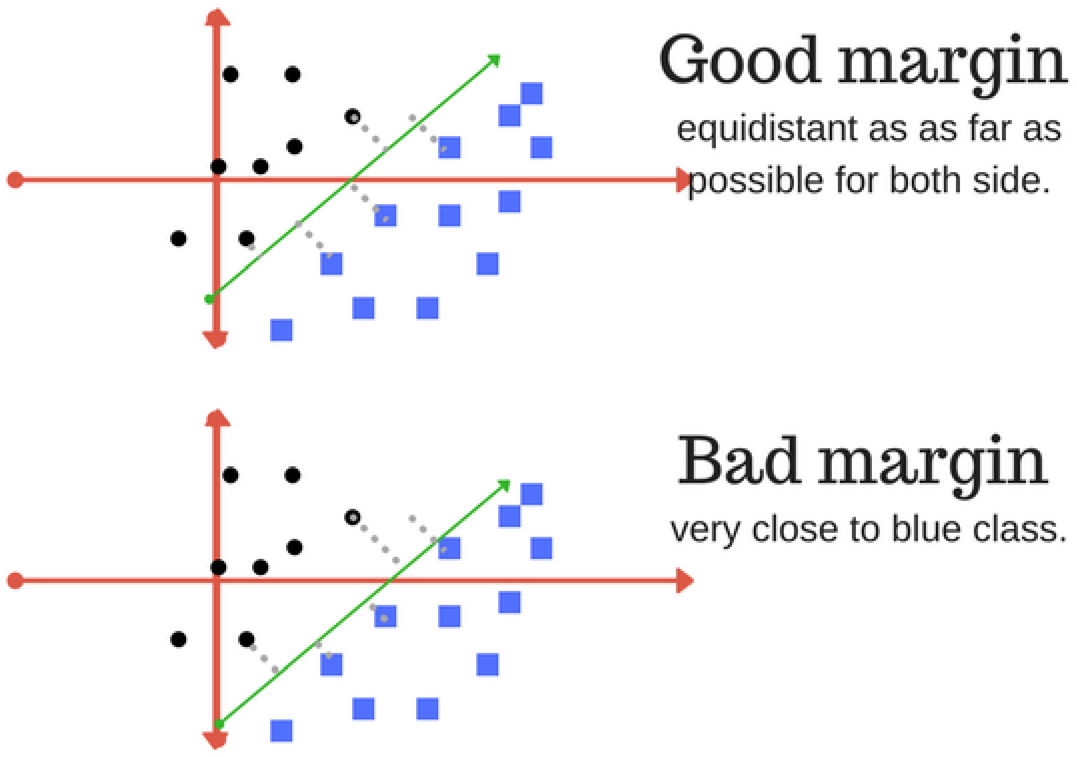

Good margin
equidistant as as far as possible for both side.
Bad margin
very close to blue class.

Figure 2.6: Graph with good and bad MARGIN

## 2.2 CORPUS

Corpus is a large collection of texts. It is a body of written or spoken material upon which a linguistic analysis is based. Monolingual corpora represent only one language while bilingual corpora represent two languages. European Corpus Initiative (ECI) corpus is multilingual having 98 million words in Turkish, Japanese, Russian, Chinese, and other languages. The corpus may be composed of written language, spoken language or both. Spoken corpus is usually in the form of audio recordings. A corpus may be open or closed. An open corpus is one which does not claim to contain all data from a specific area while a closed corpus does claim to contain all or nearly all data from a particular field.

A corpus provides grammarians, lexicographers, and other interested parties with better descriptions of a language. Computer-procurable corpora allow linguists to adopt the principle of total accountability, retrieving all the occurrences of a particular word or structure for inspection or randomly selected samples. Corpus analysis provide lexical information, morphosyntactic information, semantic information and pragmatic information.

## 2.3 The Network: Twitter

Selection of data source to conduct the sentiment analysis plays a significant role. Social media platforms as the data sources are broadly categorized into three general categories: blogs, micro-blogging sites, and review site. Among all categories, a micro-blogging site such as Twitter has gained higher popularity due to its limited strength of the content and publically availability of data. From the following statistics of the Twitter growth rate, it’s evident to use Twitter as the data source for sentiment analysis.

Twitter Growth Rate Statistics: Approximately 6,000 tweets are tweeted on Twitter on per second basis. It resembles to 350,000 tweets sent per minute and 500 million tweets per day. That makes it around 200 billion tweets per year. In Twitter's history, the number of Tweets increased from 5,000 tweets per day in 2007 to 500,000,000 tweets per day in 2013, which is approximately a six orders of magnitude. At the intermediate stages it has the statistics of 300,000 tweets per day in 2008, 2.5 million tweets per day in 2009, 35 million tweets per day in 2010, 200 million tweets per day in 2011. And 340 million tweets per day six years after the emergence of Twitter i.e. on March 21, 2012. This statistics conclude the use of Twitter for our research.

The message posted on Twitter is called Tweet, which is limited to 140 characters. Tweets are generally composed of one of the followings: text, links, International Conference on Computing, Communication and Automation (ICCCA2016) 151 emoticons, and images. A six seconds video is even added as a Tweet component in 2012. Based on these components the mining is applied to classify text, links, images, emoji or emoticons and even videos. The Tweets contains three notations including hashtags (#), retweets (RT) and account Id (@).

## 2.4 Streaming with Tweepy

Tweepy makes it easier to use the twitter streaming API by handling authentication, connection, creating and destroying the session, reading incoming messages, and partially routing messages. This page aims to help you get started using Twitter streams with Tweepy by offering a first walk through. Some features of Tweepy streaming are not covered here. See streaming.py in the Tweepy source code.

API authorization is required to access Twitter streams.

The Twitter streaming API is used to download twitter messages in real time. It is useful for obtaining a high volume of tweets, or for creating a live feed using a site stream or user stream. See the Twitter Streaming API Documentation. The streaming API is quite different from the REST API because the REST API is used to pull data from twitter but the streaming API pushes messages to a persistent session. This allows the streaming API to download more data in real time than could be done using the REST API. In Tweepy, an instance of tweepy. Stream establishes a streaming session and routes messages to StreamListener instance. The on_data method of a stream listener receives all messages and calls functions according to the message type. The default StreamListener can classify most common twitter messages and routes them to appropriately named methods, but these methods are only stubs.

Therefore using the streaming API has three steps.

1. Create a class inheriting from StreamListener
2. Using that class create a Stream object
3. Connect to the Twitter API using the Stream.

## 2.5 Parts of speech tagging

Part-of-speech tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context—i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph. A simplified form of this is commonly taught to school-age children, in the identification of words as nouns, verbs, adjectives, adverbs, etc.

Once performed by hand, POS tagging is now done in the context of computational linguistics, using algorithms which associate discrete terms, as well as hidden parts of speech, in accordance with a set of descriptive tags. POS-tagging algorithms fall into two distinctive groups: rule-based and stochastic. E. Brill's tagger, one of the first and most widely used English POS-taggers, employs rule-based algorithms.

## 2.6 Subjectivity

Subjectivity refers to how someone’s judgment is shaped by personal opinions and feelings instead of outside influences. *Subjectivity* is partially responsible for why one person loves an abstract painting while another person hates it.

Since a subject is a person, subjectivity refers to how a person's own uniqueness influences their perceptions. For example, if you have six sisters, that might influence how you view women or families — it's part of your subjectivity. Subjectivity is a form of bias and also individuality. Subjectivity is the opposite of objectivity, which is based purely on the facts and isn't personal. We expect judges to put aside their subjectivity and make decisions based on objectivity.

## 2.7 Objectivity

The terms “objectivity” and “subjectivity,” in their modern usage, generally relate to a perceiving subject (normally a person) and a perceived or unperceived object. The object is something that presumably exists independent of the subject’s perception of it. In other words, the object would be there, as it is, even if no subject perceived it. Hence, objectivity is typically associated with ideas such as reality, truth and reliability.

The perceiving subject can either perceive accurately or seem to perceive features of the object that are not in the object. For example, a perceiving subject suffering from jaundice could seem to perceive an object as yellow when the object is not actually yellow. Hence, the term “subjective” typically indicates the possibility of error.

The potential for discrepancies between features of the subject’s perceptual impressions and the real qualities of the perceived object generates philosophical questions. There are also philosophical questions regarding the nature of objective reality and the nature of our so-called subjective reality. Consequently, we have various uses of the terms “objective” and “subjective” and their cognates to express possible differences between objective reality and subjective impressions. Philosophers refer to perceptual impressions themselves as being subjective or objective. Consequent judgments are objective or subjective to varying degrees, and we divide reality into objective reality and subjective reality. Thus, it is important to distinguish the various uses of the terms “objective” and “subjective.”

## 2.8 Data Framework:

Dash is a Python Framework for building analytical web applications. It is build on top of Plotly.js, React to provide rich interactive graphing and user interfaces and Python's flask to provide a simple but scalable web server. Dash ties ample of modern UI elements. While Dash apps are viewed in the web browser, we don’t have to write any JavaScript or HTML. Dash provides a Python interface to a rich set of interactive web-based components.
Dash can be downloaded today from Python’s package manager with pip install dash—it’s entirely open-source and MIT licensed.

### 2.9 Flask and React

Dash applications are web servers running Flask and communicating JSON packets over HTTP requests. Dash’s frontend renders components using React.js, the Javascript user-interface library written and maintained by Facebook.

Flask is great. It’s widely adopted by the Python community and deployed in production environments everywhere. The underlying instance of Flask and all of its configurable properties is accessible to Dash app developers. For advanced developers, Dash apps can be extended through the rich set of Flask Plugins as well.

React is fantastic too. At Plotly, we’ve rewritten our entire web-platform and our online chart editor with React. One of the incredible things about React is how

prolific and talented the community is. The open source React community has published thousands of high quality interactive components,
from Dropdowns to Sliders to Calendar Pickers to Interactive Tables.

Dash leverages the power of Flask and React, putting them to work for Python data scientists who may not be expert Web programmers.

### 2.10 Data Visualization

Dash comes with a Graph component that renders charts with plotly.js. Plotly.js is a great fit for Dash: it's declarative, open source, fast, and supports a complete range of scientific, financial, and business charts. Dash's Graph element shares the same syntax as the open-source plotly.py library. Dash's Graph component hooks into the plotly.js event system, allowing Dash app authors to write applications that respond to hovering, clicking, or selecting points on a Plotly graph.

## 2.11 Data Collection – Twitter API

For analyzing the live data, now-a-days, there is no better option than Twitter sentiments. We have chosen Twitter data as our project input. For this purpose we make use of API's twitter provider. Twitter provides 2 API's. They are Stream and REST API's. The main difference between them is that streaming supports long lived connections and provides data in almost real-time. Whereas REST API connections are short lived and are rate limited. (One can download only a certain amount of data but not more than that). Streaming API allow access to twitter data such as status updates and user information regardless of time. However, Twitter does not provide the past data which is one week older.

Streaming API allow access to these live data and can store the data in our local file system for further uses.  For the need of our project we have used Twitter's Streaming API.

## 2.12 Preprocessing

The tweets gathered from the twitter are a mixture of

Language Detection- We are mainly interested in English texts only. All Tweets have been separated into English. We don’t consider the non-English ones. This is possible using NLTK language detection feature.

Stop Words- In information retrieval, it is a common tactic to ignore very common words such as “a” “an” “the” etc. Since their appearance in the tweet does not provide any useful information in classifying a document.

Stemming- For grammatical reasons, documents are going to use different forms of a word, such as *organize*, *organizes*, and *organizing*. Additionally, there are families of derivationally related words with similar meanings, such as *democracy*, *democratic*, and *democratization*. In many situations, it seems as if it would be useful for a search for one of these words to return documentsthat contain another word in the set. The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For Example: am, is, are is converted into be etc.

Strip Smileys-   Many microblogging posts make use of smiles to convey emotions, making them very useful in TEXT analysis for sentiments. Almost all the smiles and emotions are stripped in our project.

Erratic Casting:- In order to address the problem of posts containing  various casing of words (eg. “HeLLo”) we sanitize the input by lowering cases all words which provides some consistence in the lexicon.

# 3. LITERATURE REVIEW

Efthymios Kouloumpis, TheresaWilson, Johns Hopkins University, USA, Johanna Moore,School of Informatics University of Edinburgh, Edinburgh, UK in a paper on Twitter Sentiment Analysis: **The Good the Bad and the OMG**! in july 2011 have investigated the utility of linguistic features for detecting the sentiment of Twitter messages. We evaluate the usefulness of existing lexical resources as well as features that capture information about the informal and creative language used in microblogging. We take a supervised approach to the platform, but leverage existing hashtags in the Twitter data for building training data.

Subhabrata Mukherjee, Akshat Malu, balamurali A.R.12, Pushpak Bhattacharyya, Dept.of Computer Science and Engineering, IIT Bombay, IITB-Monash Research Academy, IIT Bombay on a paper on TwiSent: **A Multistage System for Analyzing Sentiment in Twitter** in Feb 2013 they have presented TwiSent, a sentiment analysis system for Twitter. Based on the Topic searched, Twisent collects tweets pertainign to it and categories them into the different polarity classes posotive, negative and objective. However, analyzing micro-blog posts have many inherent challenges compared to the other text genres.

# 4. Methodology &Work Description

## FLOW CHART

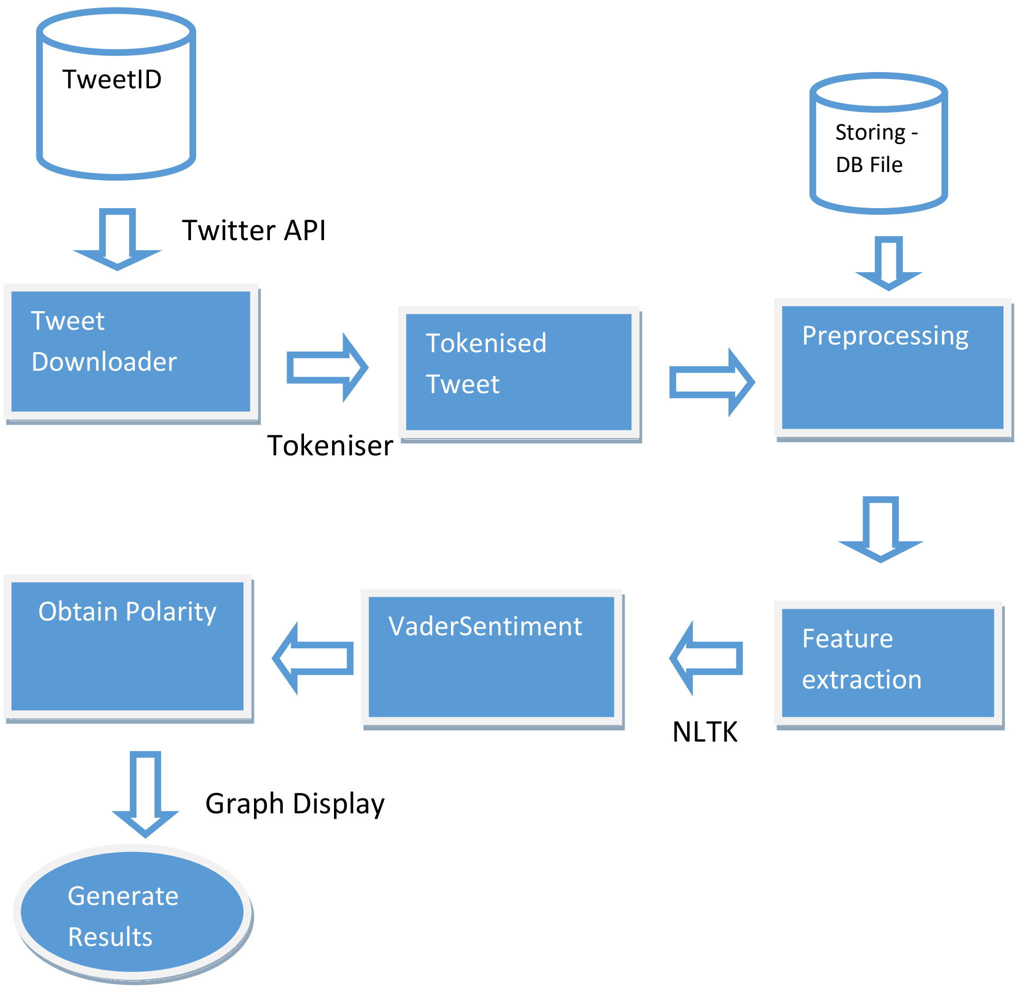


Figure 4.1: FlowChart

## Work Flow

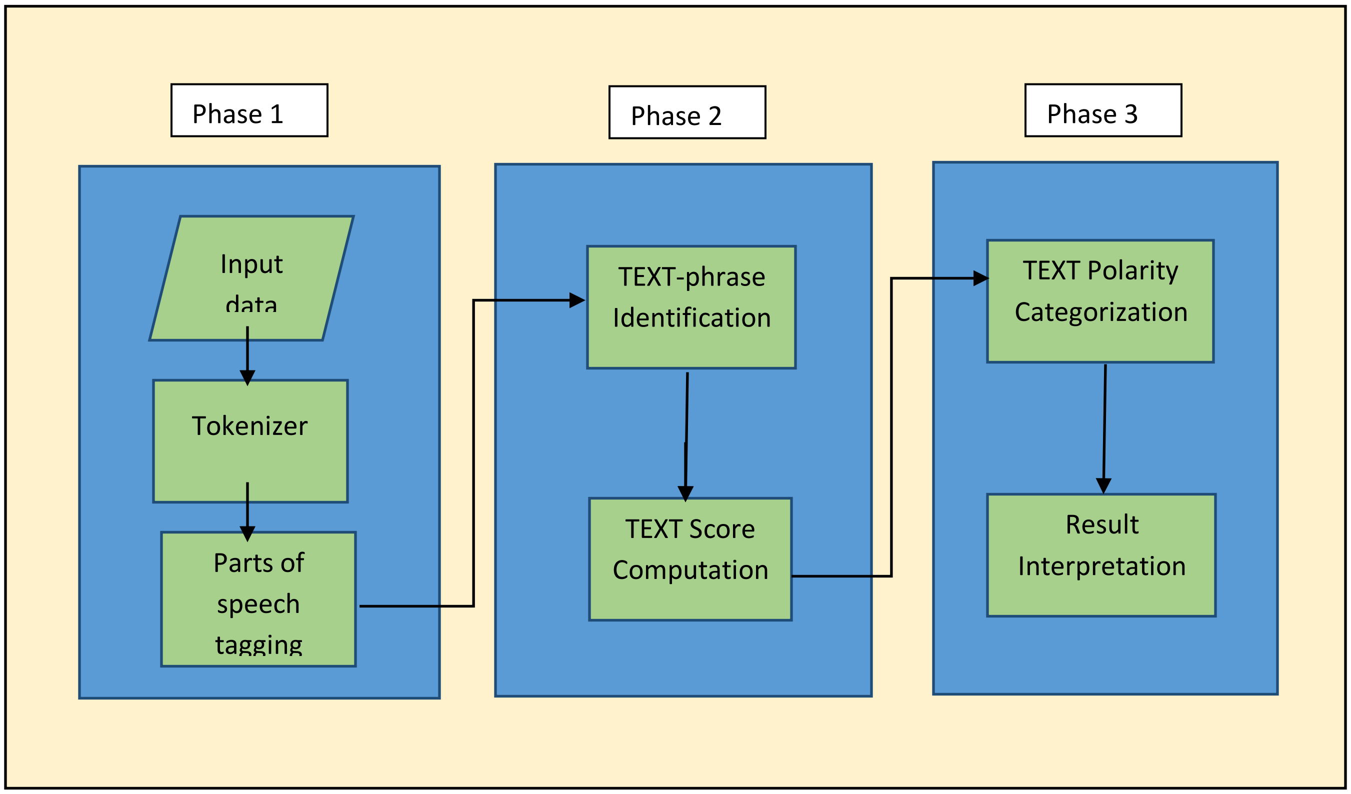


Figure 4.2: Workflow

We have divided the entire work into five different modules.

1) Getting live data from twitter- The first step is to collect the data from the the Twitter development website https://dev.twitter.com/docs . The data is collected by using the unique keys and tokens assigned by twitter website .These are then implemented onto the code by storing them using the ‘tweepy’ library in python.

2) Processing the tweet- The tweets now generated contains lot of headers and unwanted data that would be required to be processed using Natural Language Processing Toolkit. The different steps carried out here are stop-word removal, stemming, case lowering etc.

3) Calculating the sentiment- The processed tweets are then sent through a library called Vader sentiment. This library is built on the concept of SVM. This library takes in the tweets are generates the sentiment classified into positive, neutral and neutral readily.

4) Visualization in dash- The results generated are shown in a framework called Dash. Dash is a framework that is used to display the visual results with optimum quality and varied options ex. Capture, graph comparing etc.

5) Word cloud-The tweets generated are finally represented using word cloud to depict the current trending topics and to give a pictorial representation of the usage of various key words in the generated tweets.

# 5. Tools and technologies

## 5.1 Software Requirements:

- Python Compiler (any version above 2.7.0)
- Dash Framework dependencies.
- NLTK Tool Kit
- Libraries consisting of Word cloud, Vader Sentiment, StopWordsRemoval etc.
- Twitter API Streaming
- A Good Web Browser
- Any good Operating System Such as LINUX, WINDOWS, MAC etc.

## 5.2 Hardware Requirements:

- 2GB RAM
- i3 Processor

# 6. Implementation & Code

- CODE FOR ANALYSIS OF TWEETS

```
 import dash
from dash.dependencies import Output, Event, Input
import dash_core_components as dcc
import dash_html_components as html
import plotly
import random
import plotly.graph_objs as go
from collections import deque
import sqlite3
import pandas as pd
import time
#popular topics: google, olympics, trump, gun, usa
app = dash.Dash(__name__)
app.layout = html.Div(
[   html.H2('Live Twitter Sentiment'),
dcc.Input(id='sentiment_term', value='google', type='text'),
dcc.Graph(id='live-graph', animate=False),
dcc.Interval(id='graph-update',interval=1*1000),])
```

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

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

 events=[Event('graph-update', 'interval')])

 def update_graph_scatter(sentiment_term):

 try:conn = sqlite3.connect('twitter.db')

 c = conn.cursor()

 df = pd.read_sql("SELECT * FROM sentiment WHERE tweet LIKE ? ORDER BY
 unix DESC LIMIT 1000", conn ,params=('%' + sentiment_term + '%',))

 df.sort_values('unix', inplace=True)

 df['sentiment_smoothed'] = df['sentiment'].rolling(int(len(df)/5)).mean()

df['date'] = pd.to_datetime(df['unix'],unit='ms')

 df.set_index('date', inplace=True)

df = df.resample('1*s').mean()

df.dropna(inplace=True)

#X = df.unix.values[-100:]

 X = df.index[-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)]),

 title='Term: {}'.format(sentiment_term))}
```

```
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=True)
```

- CODE FOR GENERATING LIVE TWEETS

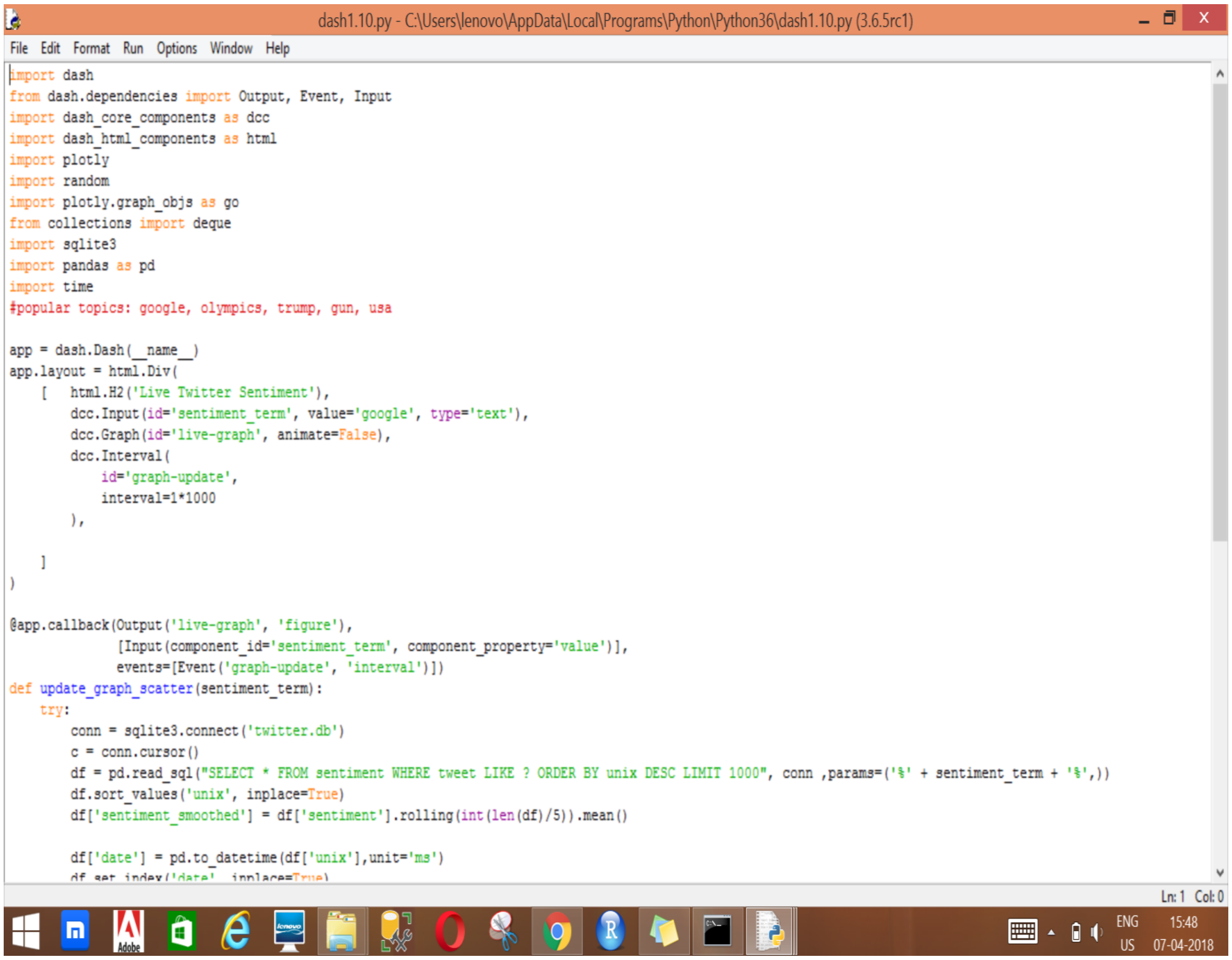
```
 from tweepy import Stream
 from tweepy import OAuthHandler
 from tweepy.streaming import StreamListener
 import json
 import sqlite3
 from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
 from unidecode import unidecode
 import timeanalyzer = SentimentIntensityAnalyzer()

ckey="EHVQjHUcpXM9FeAgHjoqoc2p0"

 csecret="qPBuGsWiBecUiI9g1c9ZcF4Sn3RFBb3G8Ct26h8JovX7Pc2xba"
 atoken="975958509857705989-aOCUwXelJ1CCLRgUyPxGePTczYBaqTW"
 asecret="uEr18dx9u6J6jYTtlkB14o6XVx7huP8DfRNEJFHjDvaH2"

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

 c = conn.cursor()
 #c.execute("DROP TABLE sentiment")

def create_table():

    try:
        c.execute("CREATE TABLE IF NOT EXISTS sentiment(unix REAL, tweet
 TEXT, sentiment REAL)")
        c.execute("CREATE INDEX fast_unix ON sentiment(unix)")
        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:
            data = json.loads(data)
            tweet = unidecode(data['text'])
            time_ms = data['timestamp_ms']
            vs = analyzer.polarity_scores(tweet)
            sentiment = vs['compound']
            print(time_ms, tweet, sentiment)
            f = open("data.txt","a+")




            f.write(tweet)
            c.execute("INSERT INTO sentiment (unix, tweet, sentiment)
VALUES (?, ?, ?)",
                  (time_ms, tweet, sentiment))
            conn.commit()

        except KeyError as e:
            print(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())
        twitterStream.filter(track=["a","e","i","o","u"])
```

```
except Exception as e:
    print(str(e))
    time.sleep(5)
```

- CODE FOR WORD CLOUD

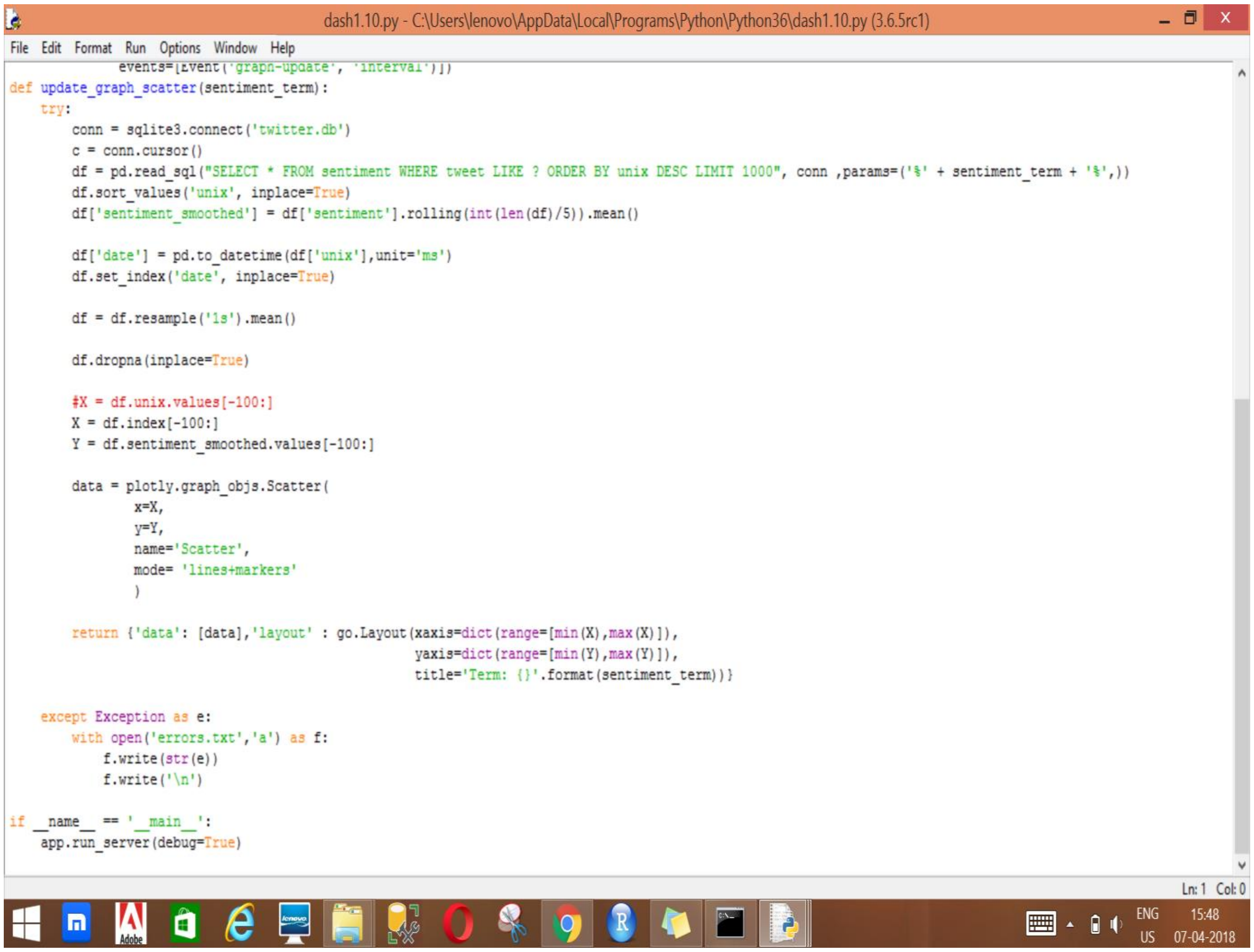
```
 from wordcloud import WordCloud, STOPWORDS

 import matplotlib.pyplot as plt

 from PIL import Image

 import numpy as np

 f = open("data.txt", "r")

 text = f.read()

our_mask = np.array(Image.open('wc-mask3.png'))

stopwords = set(STOPWORDS)

 stopwords.add("https")

 stopwords.add("co")

cloud = WordCloud(background_color="black", mask = our_mask,

 stopwords = stopwords).generate(text)

plt.imshow(cloud)

 plt.axis('off')

 plt.show()
```

# 7. Result and Analysis


dash1.10.py - C:\Users\lenovo\AppData\Local\Programs\Python\Python36\dash1.10.py (3.6.5rc1)
File Edit Format Run Options Window Help
Ln: 1 Col: 0
ENG US 15:48 07-04-2018

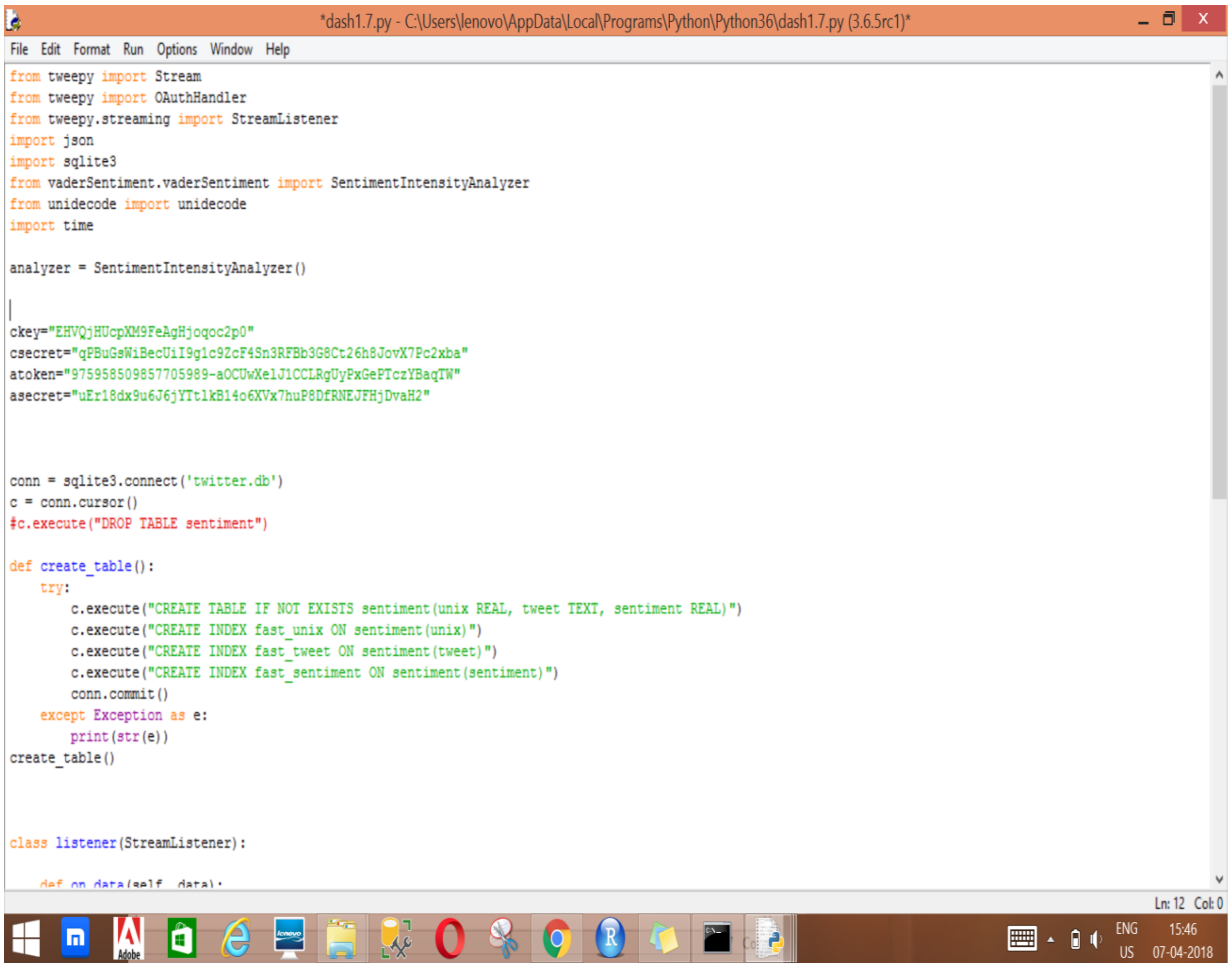
```
import dash
from dash.dependencies import Output, Event, Input
import dash_core_components as dcc
import dash_html_components as html
import plotly
import random
import plotly.graph_objs as go
from collections import deque
import sqlite3
import pandas as pd
import time
#popular topics: google, olympics, trump, gun, usa

app = dash.Dash(__name__)
app.layout = html.Div(
    [   html.H2('Live Twitter Sentiment'),
        dcc.Input(id='sentiment_term', value='google', type='text'),
        dcc.Graph(id='live-graph', animate=False),
        dcc.Interval(
            id='graph-update',
            interval=1*1000
        ),

    ]
)

@app.callback(Output('live-graph', 'figure'),
              [Input(component_id='sentiment_term', component_property='value')],
              events=[Event('graph-update', 'interval')])
def update_graph_scatter(sentiment_term):
    try:
        conn = sqlite3.connect('twitter.db')
        c = conn.cursor()
        df = pd.read_sql("SELECT * FROM sentiment WHERE tweet LIKE ? ORDER BY unix DESC LIMIT 1000", conn ,params=('%' + sentiment_term + '%',))
        df.sort_values('unix', inplace=True)
        df['sentiment_smoothed'] = df['sentiment'].rolling(int(len(df)/5)).mean()

        df['date'] = pd.to_datetime(df['unix'],unit='ms')
        df.set_index('date', inplace=True)
```

Figure 7.1: dash framework

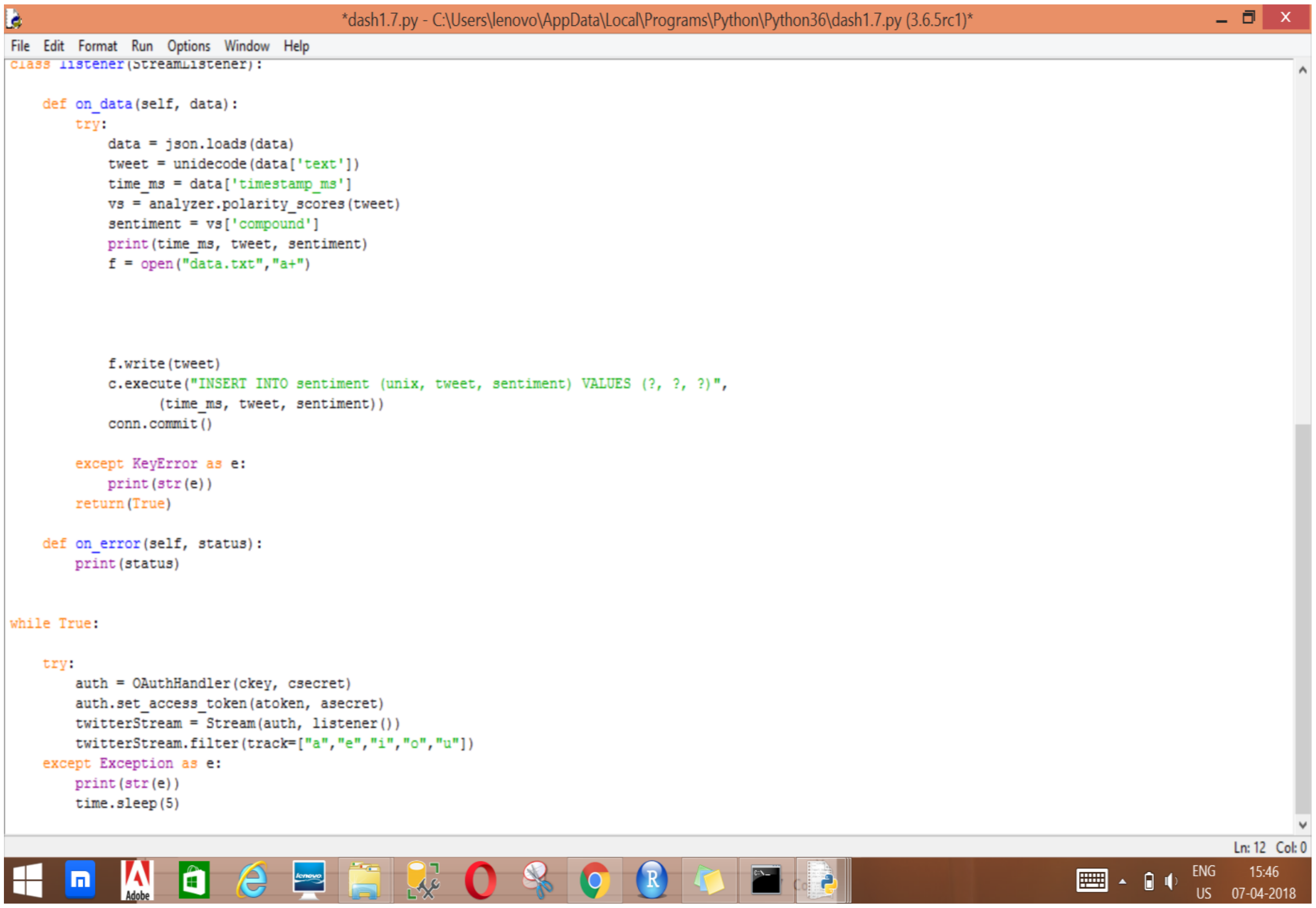


dash1.10.py - C:\Users\lenovo\AppData\Local\Programs\Python\Python36\dash1.10.py (3.6.5rc1)
File Edit Format Run Options Window Help

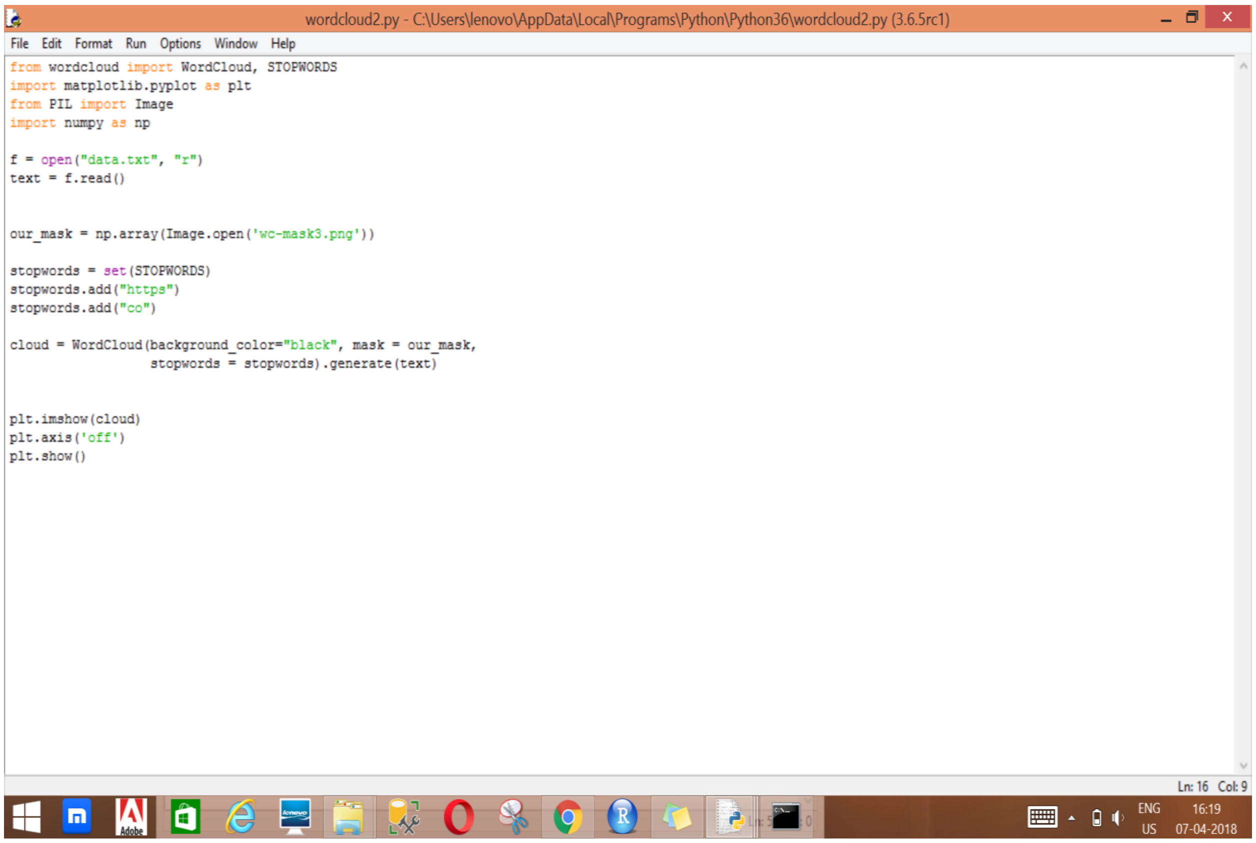
```
            events=[Event('graph-update', 'interval')])
def update_graph_scatter(sentiment_term):
    try:
        conn = sqlite3.connect('twitter.db')
        c = conn.cursor()
        df = pd.read_sql("SELECT * FROM sentiment WHERE tweet LIKE ? ORDER BY unix DESC LIMIT 1000", conn ,params=('%' + sentiment_term + '%',))
        df.sort_values('unix', inplace=True)
        df['sentiment_smoothed'] = df['sentiment'].rolling(int(len(df)/5)).mean()

        df['date'] = pd.to_datetime(df['unix'],unit='ms')
        df.set_index('date', inplace=True)

        df = df.resample('1s').mean()

        df.dropna(inplace=True)

        #X = df.unix.values[-100:]
        X = df.index[-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)]),
                                                    title='Term: {}'.format(sentiment_term))}

    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=True)
```


*dash1.7.py - C:\Users\lenovo\AppData\Local\Programs\Python\Python36\dash1.7.py (3.6.5rc1)*
File Edit Format Run Options Window Help
Ln: 12 Col: 0
ENG US 15:46 07-04-2018

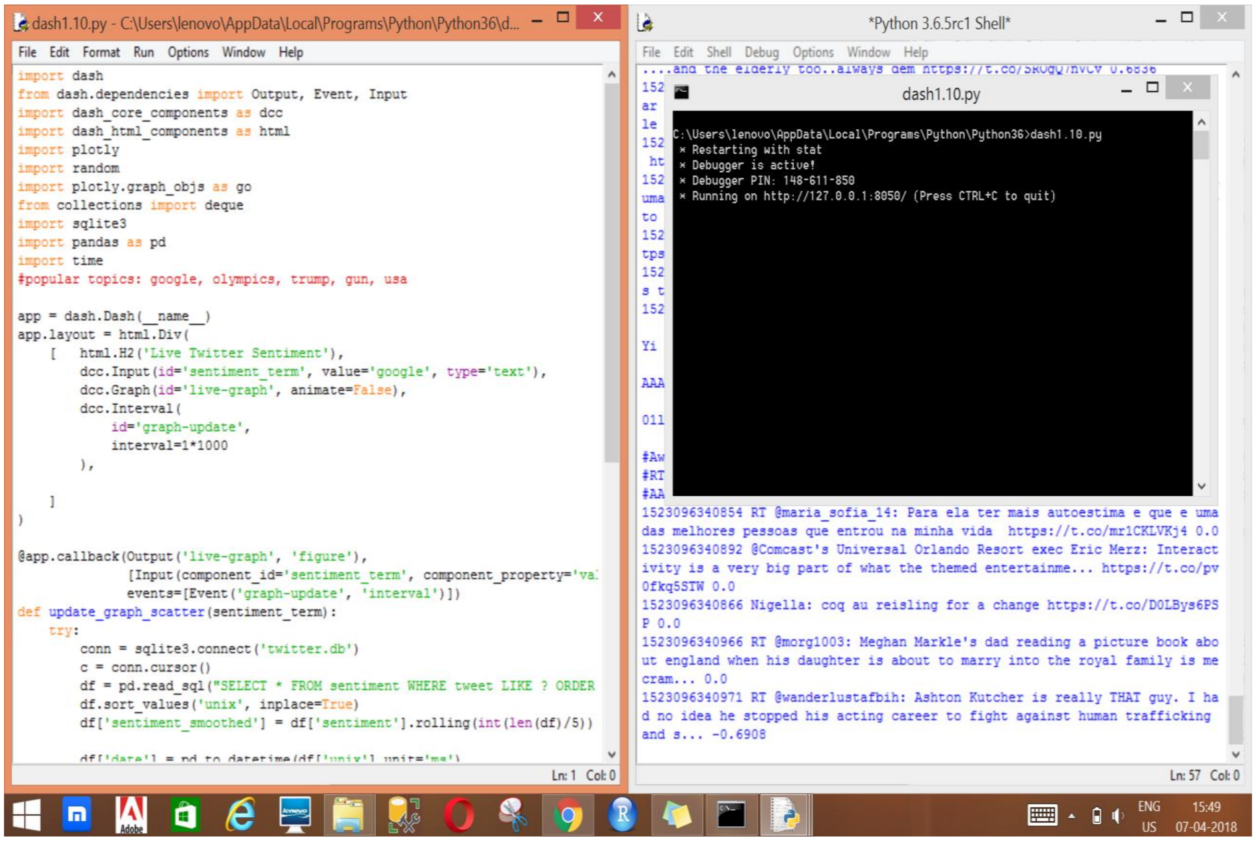
```
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import json
import sqlite3
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from unidecode import unidecode
import time

analyzer = SentimentIntensityAnalyzer()


ckey="EHVQjHUcpXM9FeAgHjoqoc2p0"
csecret="qPBuGsWiBecUiI9g1c9ZcF4Sn3RFBb3G8Ct26h8JovX7Pc2xba"
atoken="975958509857705989-aOCUwXe1J1CCLRgUyPxGePTczYBaqTW"
asecret="uEr18dx9u6J6jYTtlkB14o6XVx7huP8DfRNEJFHjDvaH2"



conn = sqlite3.connect('twitter.db')
c = conn.cursor()
#c.execute("DROP TABLE sentiment")

def create_table():
    try:
        c.execute("CREATE TABLE IF NOT EXISTS sentiment(unix REAL, tweet TEXT, sentiment REAL)")
        c.execute("CREATE INDEX fast_unix ON sentiment(unix)")
        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):
```


*dash1.7.py - C:\Users\lenovo\AppData\Local\Programs\Python\Python36\dash1.7.py (3.6.5rc1)*
File Edit Format Run Options Window Help
Ln: 12 Col: 0
ENG US 15:46 07-04-2018

```
class listener(StreamListener):

    def on_data(self, data):
        try:
            data = json.loads(data)
            tweet = unidecode(data['text'])
            time_ms = data['timestamp_ms']
            vs = analyzer.polarity_scores(tweet)
            sentiment = vs['compound']
            print(time_ms, tweet, sentiment)
            f = open("data.txt","a+")




            f.write(tweet)
            c.execute("INSERT INTO sentiment (unix, tweet, sentiment) VALUES (?, ?, ?)",
                  (time_ms, tweet, sentiment))
            conn.commit()

        except KeyError as e:
            print(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())
        twitterStream.filter(track=["a","e","i","o","u"])
    except Exception as e:
        print(str(e))
        time.sleep(5)
```


wordcloud2.py - C:\Users\lenovo\AppData\Local\Programs\Python\Python36\wordcloud2.py (3.6.5rc1)
File Edit Format Run Options Window Help
Ln: 16 Col: 9
ENG US 16:19 07-04-2018

```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np

f = open("data.txt", "r")
text = f.read()


our_mask = np.array(Image.open('wc-mask3.png'))

stopwords = set(STOPWORDS)
stopwords.add("https")
stopwords.add("co")

cloud = WordCloud(background_color="black", mask = our_mask,
                  stopwords = stopwords).generate(text)


plt.imshow(cloud)
plt.axis('off')
plt.show()
```



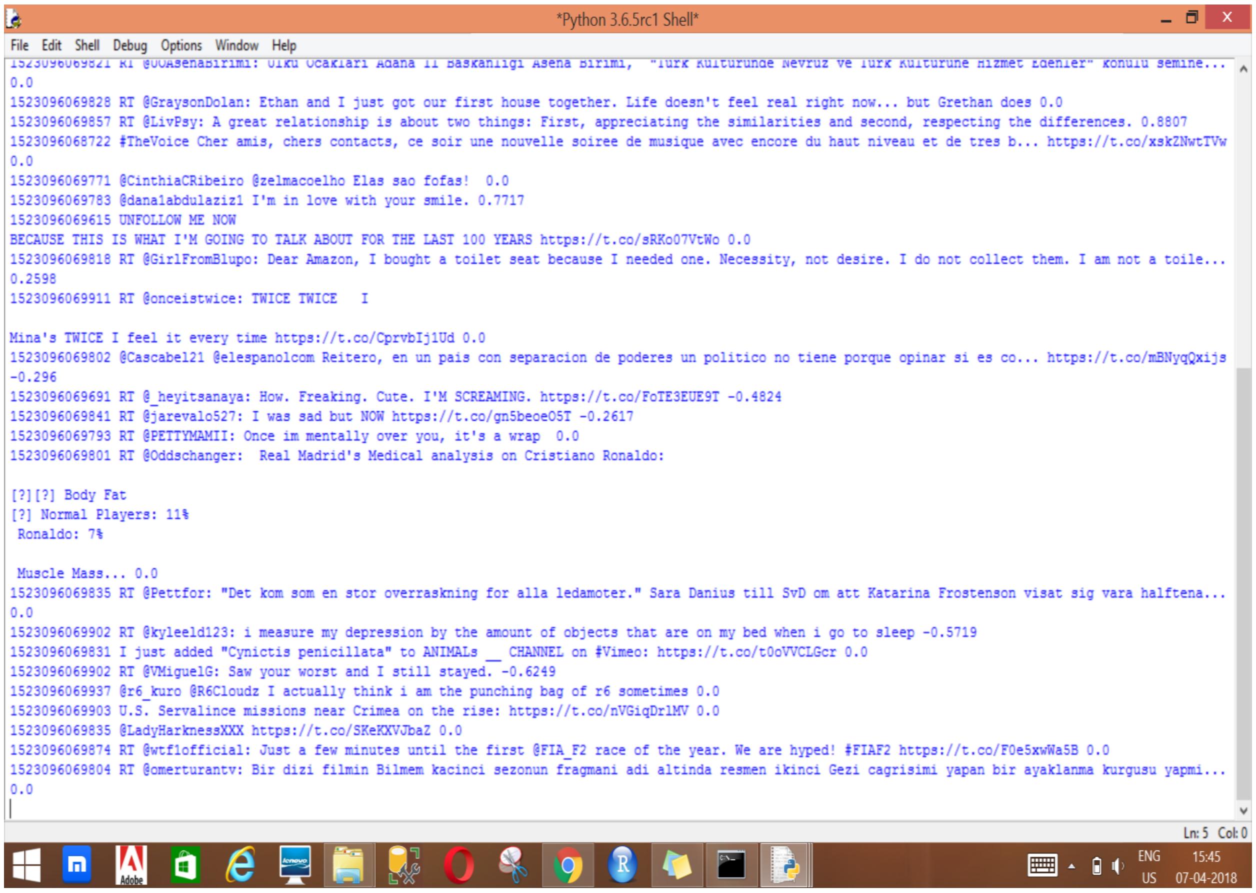
```
import dash
from dash.dependencies import Output, Event, Input
import dash_core_components as dcc
import dash_html_components as html
import plotly
import random
import plotly.graph_objs as go
from collections import deque
import sqlite3
import pandas as pd
import time
#popular topics: google, olympics, trump, gun, usa

app = dash.Dash(__name__)
app.layout = html.Div(
    [   html.H2('Live Twitter Sentiment'),
        dcc.Input(id='sentiment_term', value='google', type='text'),
        dcc.Graph(id='live-graph', animate=False),
        dcc.Interval(
            id='graph-update',
            interval=1*1000
        ),

    ]
)

@app.callback(Output('live-graph', 'figure'),
              [Input(component_id='sentiment_term', component_property='va
              events=[Event('graph-update', 'interval')])
def update_graph_scatter(sentiment_term):
    try:
        conn = sqlite3.connect('twitter.db')
        c = conn.cursor()
        df = pd.read_sql("SELECT * FROM sentiment WHERE tweet LIKE ? ORDER
        df.sort_values('unix', inplace=True)
        df['sentiment_smoothed'] = df['sentiment'].rolling(int(len(df)/5))

        df['date'] = pd.to_datetime(df['unix'],unit='ms')
```

```
C:\Users\lenovo\AppData\Local\Programs\Python\Python36>dash1.10.py
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 148-611-850
 * Running on http://127.0.0.1:8050/ (Press CTRL+C to quit)
```

```
....and the elderly too..always dem https://t.co/SKUgQ7hvCv 0.6836
1523096340854 RT @maria_sofia_14: Para ela ter mais autoestima e que e uma
das melhores pessoas que entrou na minha vida  https://t.co/mr1CKLVKj4 0.0
1523096340892 @Comcast's Universal Orlando Resort exec Eric Merz: Interact
ivity is a very big part of what the themed entertainme... https://t.co/pv
0fkq5STW 0.0
1523096340866 Nigella: coq au reisling for a change https://t.co/D0LBys6PS
P 0.0
1523096340966 RT @morg1003: Meghan Markle's dad reading a picture book abo
ut england when his daughter is about to marry into the royal family is me
cram... 0.0
1523096340971 RT @wanderlustafbih: Ashton Kutcher is really THAT guy. I ha
d no idea he stopped his acting career to fight against human trafficking
and s... -0.6908
```

dash1.10.py - C:\Users\lenovo\AppData\Local\Programs\Python\Python36\d...
File Edit Format Run Options Window Help
Ln: 1 Col: 0
*Python 3.6.5rc1 Shell*
File Edit Shell Debug Options Window Help
dash1.10.py
Ln: 57 Col: 0
ENG US 15:49 07-04-2018

Figure 7.3:how server is started



```
1523096069821 RT @UUAsenaBirimi: Ulku Ocaklari Adana Il Baskanligi Asena Birimi,  "Turk Kulturunde Nevruz ve Turk Kulturune Hizmet Edenler" konulu semine...
0.0
1523096069828 RT @GraysonDolan: Ethan and I just got our first house together. Life doesn't feel real right now... but Grethan does 0.0
1523096069857 RT @LivPsy: A great relationship is about two things: First, appreciating the similarities and second, respecting the differences. 0.8807
1523096068722 #TheVoice Cher amis, chers contacts, ce soir une nouvelle soiree de musique avec encore du haut niveau et de tres b... https://t.co/xskZNwtTVw
0.0
1523096069771 @CinthiaCRibeiro @zelmacoelho Elas sao fofas!  0.0
1523096069783 @danalabdulaziz1 I'm in love with your smile. 0.7717
1523096069615 UNFOLLOW ME NOW
BECAUSE THIS IS WHAT I'M GOING TO TALK ABOUT FOR THE LAST 100 YEARS https://t.co/sRKo07VtWo 0.0
1523096069818 RT @GirlFromBlupo: Dear Amazon, I bought a toilet seat because I needed one. Necessity, not desire. I do not collect them. I am not a toile...
0.2598
1523096069911 RT @onceistwice: TWICE TWICE   I

Mina's TWICE I feel it every time https://t.co/CprvbIj1Ud 0.0
1523096069802 @Cascabel21 @elespanolcom Reitero, en un pais con separacion de poderes un politico no tiene porque opinar si es co... https://t.co/mBNyqQxijs
-0.296
1523096069691 RT @_heyitsanaya: How. Freaking. Cute. I'M SCREAMING. https://t.co/FoTE3EUE9T -0.4824
1523096069841 RT @jarevalo527: I was sad but NOW https://t.co/gn5beoe05T -0.2617
1523096069793 RT @PETTYMAMII: Once im mentally over you, it's a wrap  0.0
1523096069801 RT @Oddschanger:  Real Madrid's Medical analysis on Cristiano Ronaldo:

[?][?] Body Fat
[?] Normal Players: 11%
 Ronaldo: 7%

 Muscle Mass... 0.0
1523096069835 RT @Pettfor: "Det kom som en stor overraskning for alla ledamoter." Sara Danius till SvD om att Katarina Frostenson visat sig vara halftena...
0.0
1523096069902 RT @kyleeld123: i measure my depression by the amount of objects that are on my bed when i go to sleep -0.5719
1523096069831 I just added "Cynictis penicillata" to ANIMALs __ CHANNEL on #Vimeo: https://t.co/t0oVVCLGcr 0.0
1523096069902 RT @VMiguelG: Saw your worst and I still stayed. -0.6249
1523096069937 @r6_kuro @R6Cloudz I actually think i am the punching bag of r6 sometimes 0.0
1523096069903 U.S. Servalince missions near Crimea on the rise: https://t.co/nVGiqDrlMV 0.0
1523096069835 @LadyHarknessXXX https://t.co/SKeKXVJbaZ 0.0
1523096069874 RT @wtf1official: Just a few minutes until the first @FIA_F2 race of the year. We are hyped! #FIAF2 https://t.co/F0e5xwWa5B 0.0
1523096069804 RT @omerturantv: Bir dizi filmin Bilmem kacinci sezonun fragmani adi altinda resmen ikinci Gezi cagrisimi yapan bir ayaklanma kurgusu yapmi...
0.0
```

Figure 7.3: loading the tweets

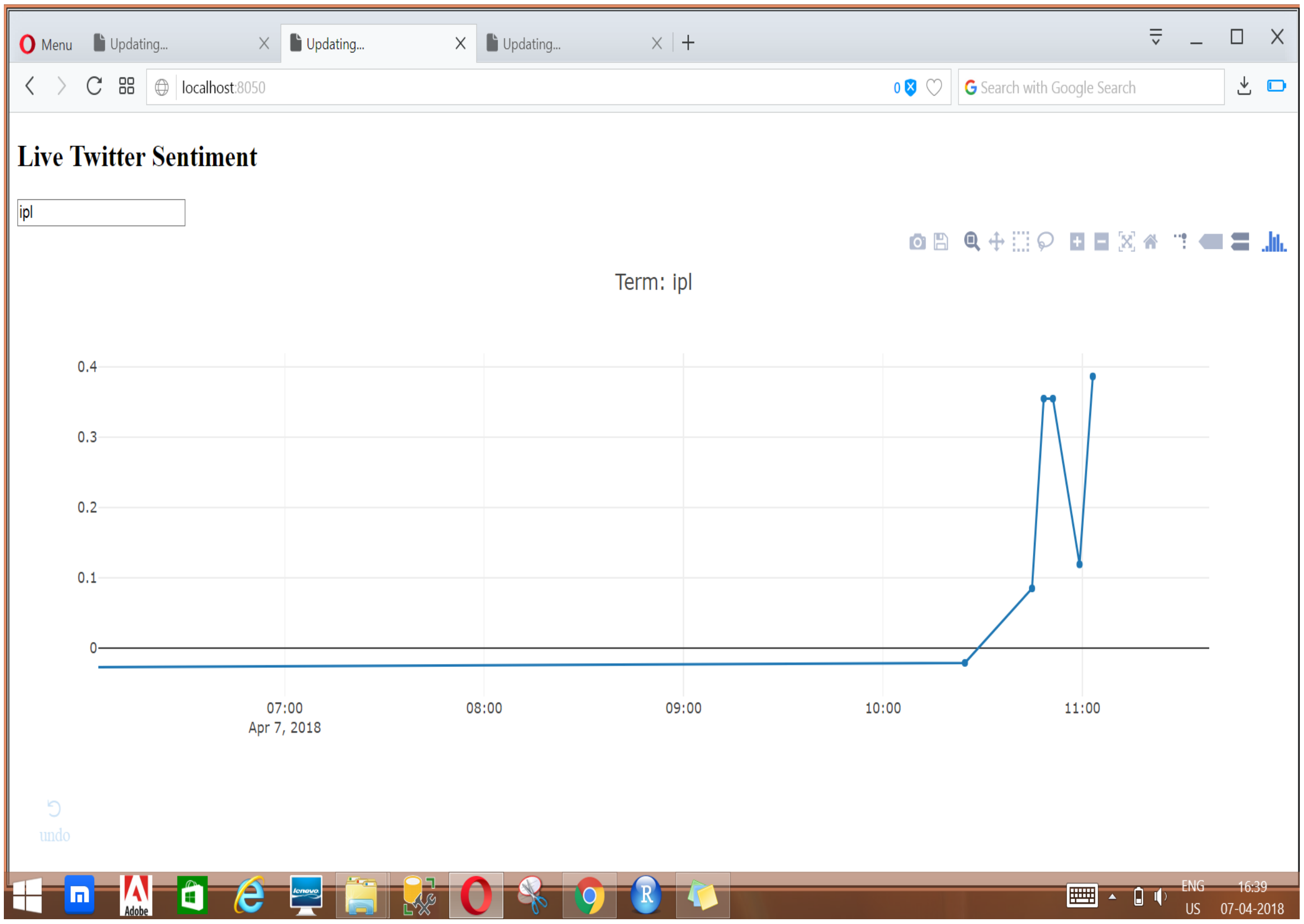


Figure 7.4: sentimental analysis for IPL

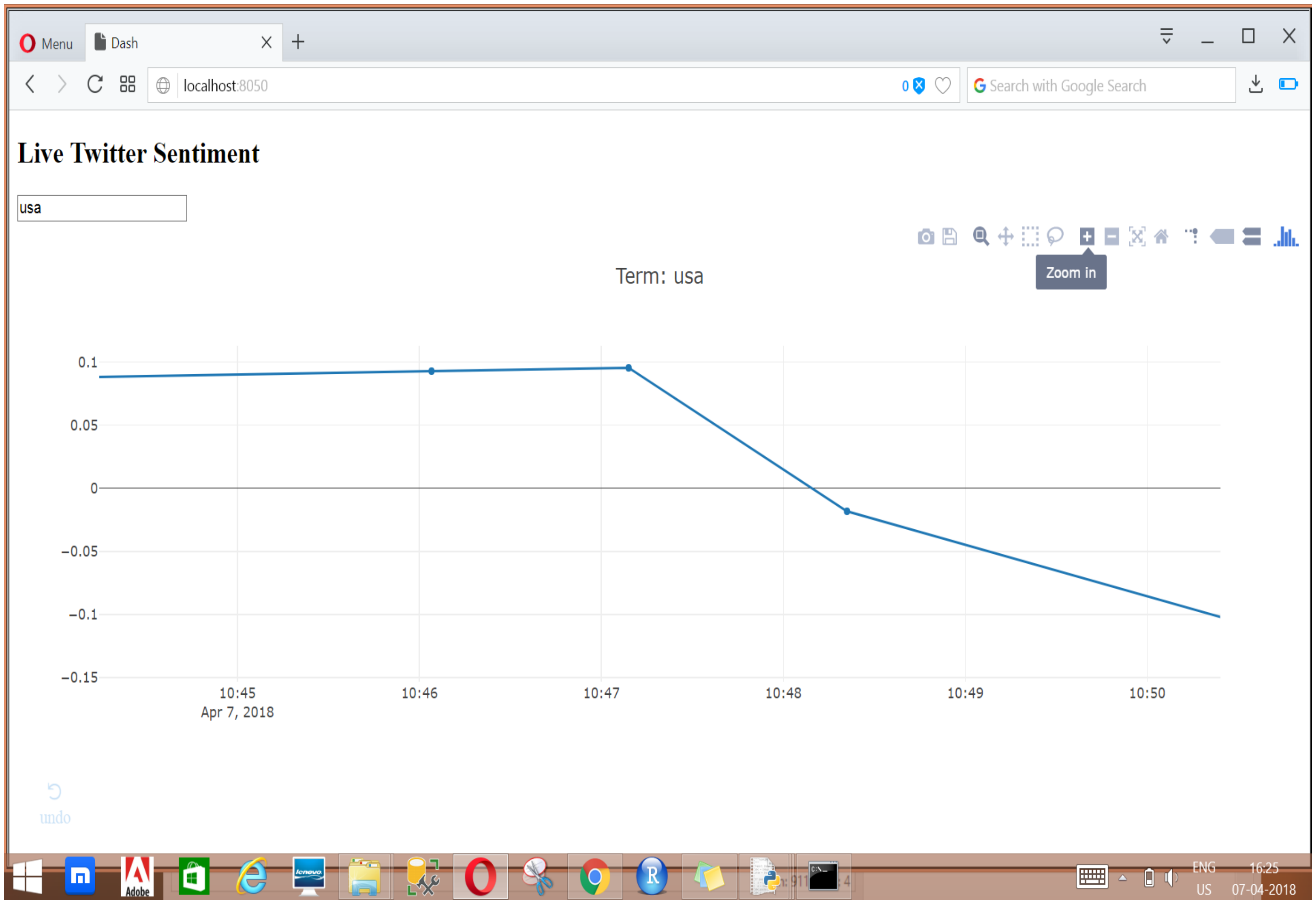


Figure 7.5: Sentiment analysis for USA

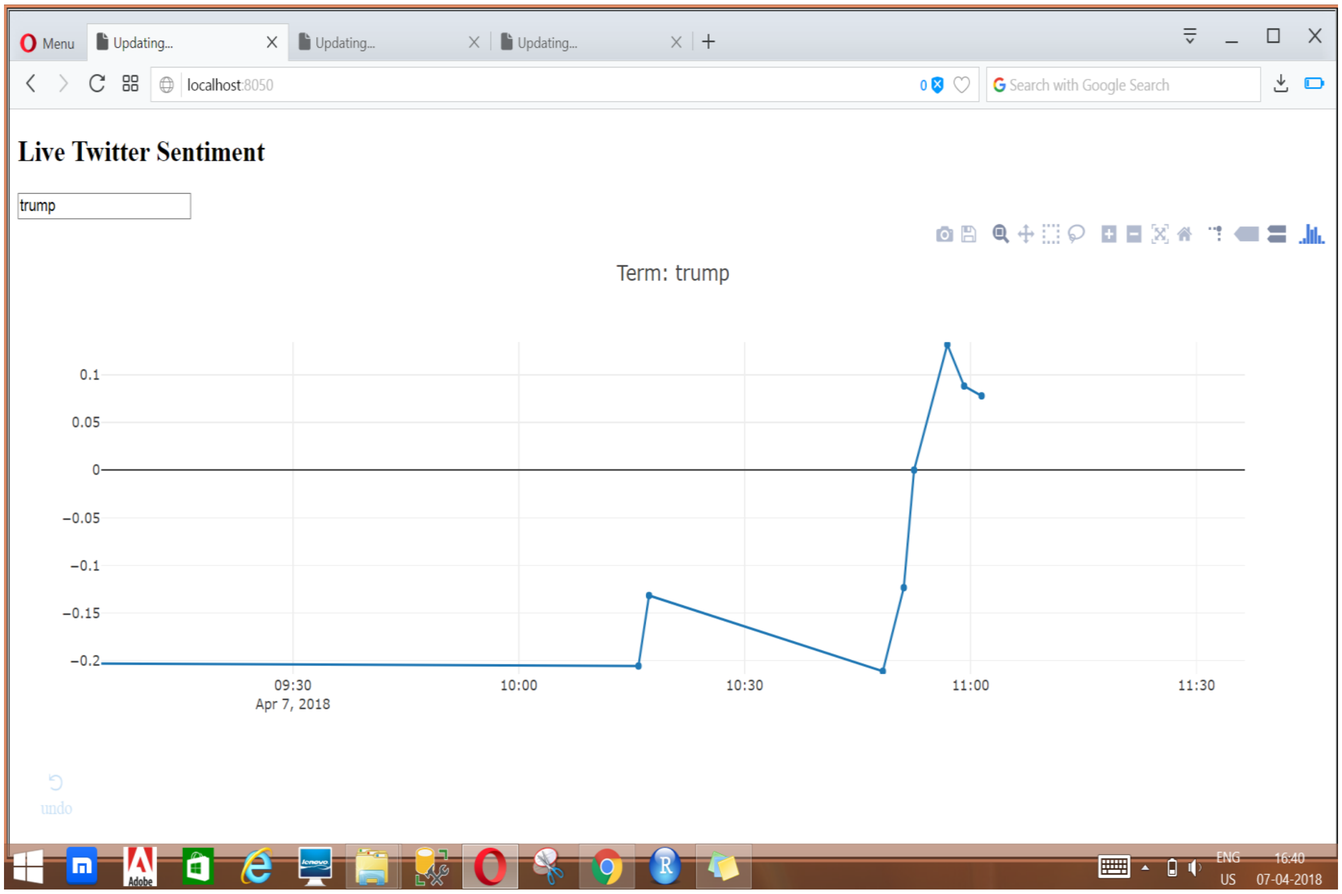


Figure 7.6: Sentiment analysis for TRUMP

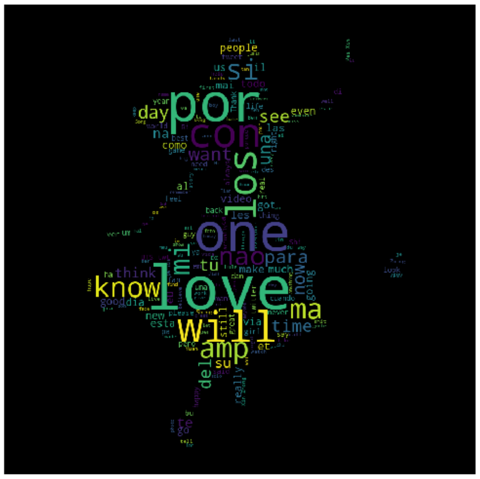


Figure 7.7: wordcloud

## 8. Expected result:

Result will be displayed in the form of graph (bar graphs, line graphs etc.,) and a word cloud giving all the insights.

The graph here used here has some points which basically represent a tweet and line emerging from that point is the sentiment of that tweet.

If the line emerging from the point moves up it means the tweet has positive sentiment and if it moves download the tweet has negative sentiment, if the line is horizontal the tweet basically has neutral sentiment.

The Word Cloud is generated here gives all the insights that are present in all the tweets that we have downloaded. Insights are basically the topics overs which the twitter is discussing.

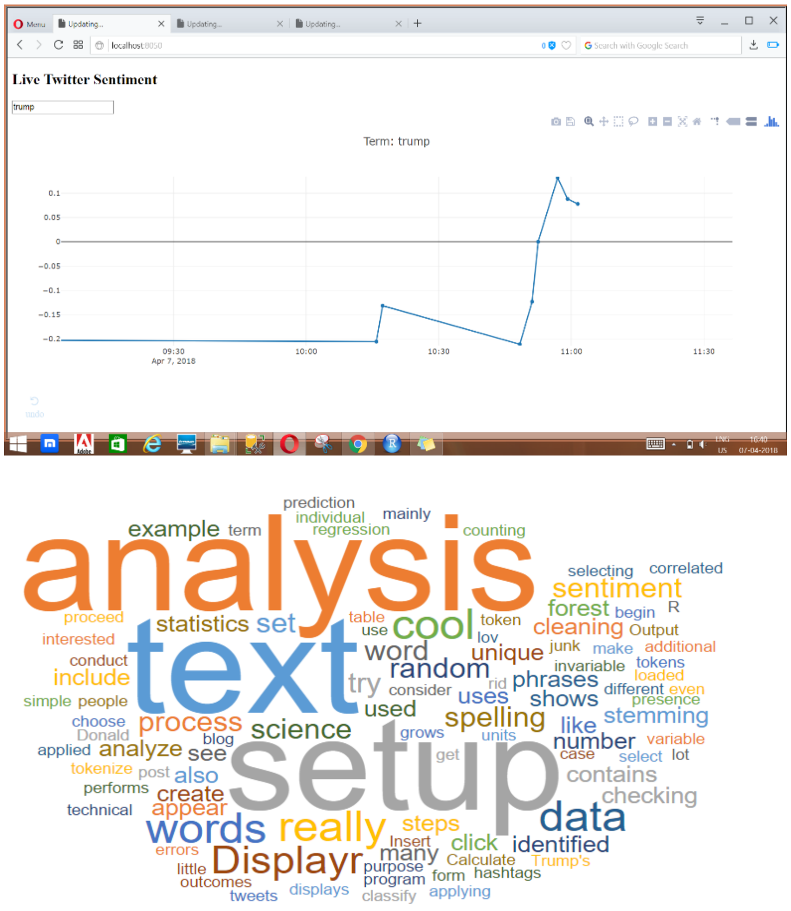


Figure 8.1: Expected Results

# 9. Conclusion & Future Scope

## Conclusion:

We conclude that using different NLTK classifiers it is easy to classify the tweets and more we improve the pre-processing of input the accurate the obtained results will be. Also if we are able to draw large number of tweets dynamically within one time frame we would be generating the output with more accuracy.

## Future Scope:

The applications for sentiment analysis are endless. More and more we’re seeing it used in social media monitoring and VOC to track customer reviews, survey responses, competitors, etc. However, it is also practical for use in business analytics and situations in which text needs to be analysed.

Sentiment analysis is in demand because of its efficiency. Thousands of text documents can be processed for sentiment (and other features including named entities, topics, themes, etc.) in seconds, compared to the hours it would take a team of people to manually complete. Because it is so efficient (and accurate – Semantria has 80% accuracy for English content) many businesses are adopting text and sentiment analysis and incorporating it into their processes.

- Computing customer satisfaction metrics
- To forecast market movement based on news, blogs and social media sentiment.
- Live Cricket win Predictor. Etc.,

## References:

[1]Lidong Wang; Guanghui Wang; Cheryl Ann Alexander, “Natural language processing systems and Big Data analytics” 2016 Vol 2 , No.2
DOI: 10.1504/IJCSYSE.2015.077052

[2]Olivera Kotevskaa, Ahmed Lbatha, Sarala Padib " Web of Things and Big Data (WoTBD 2016)", University of Grenoble Alpes, France, Indian Institute of Technology, Madras, India,

[3]Pak, A., Paroubek, P.. Twitter as a corpus for sentiment analysis and opinion mining. In: LREC; vol. 10. 2010, p. 1320–1326.

[4]Kouloumpis, E., Wilson, T., Moore, J.. Twitter sentiment analysis: The good the bad and the omg! Icwsm 2011; 11: 538–541.