PROJECT REPORT

On

Twitter Sentimental Analysis Using Python

Submitted by

Lakshay Anand

(171500173)

Department of Computer Engineering & Applications

Institute of Engineering & Technology

GLA University

Mathura-281406,INDIA

2019

Department of computer Engineering and Applications
GLA UNIVERSITY, MATHURA
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

Declaration

I hereby declare that the work which is being presented in my minor project titled"**Twitter Sentimental Analysis Using Python",** in partial fulfillmentof the requirements for my degree program, is an authentic record of my own work carried under the supervision of "Mr Pankaj Sharma Sir.

Signature of Candidate :

Name of Candidate: Lakshay Anand

Roll. No.: 171500173

Course: B.tech. (CSE)

Year: 3rd

Semester: 5th

Synopsis

B.tech.(CSE)-Batch 2017-2021

Student Information:

Name: Lakshay Anand	University Roll. No. 171500173
Mobile: 7060089177	Email: lakshay.anand_cs17@gla.ac.in

Information about Industry/Organization:

Industry/Organization Name with full Address	GLA University
Contact Person	Name & Designation: Mr Pankaj Sharma

Project Information:

Title Of Project/Training/Task	Twitter Sentimental Analysis
Role & Responsibility	
Technical Details	Software Requirements: Python,Anaconda
Implementation Details	Implemented
Period	Start Date: 1/09/2019 End Date: 20/11/2019

Summary of the Project Wor

The project titled "twitter sentimental analysis using python" is use to classify tweets of a person using classifier algorithms. First the machine is trained using some tweets data set and then the algorithm is tested over the test data sets. In this project machine is trained over 80000 tweets. When tested over the data set, it classify tweets

Acknowledgement

I thank the almighty for giving me the courage and perseverance in completing the project.

This project itself is acknowledgement for all those people who have given me their heartfelt co-operation in making this project a success.

I extend my sincere thanks to **Mr. Pankaj Sharma Sir**

for providing valuable guidance at every stage of this project work. I am profoundly grateful towards the unmatched services rendered by him.

Last but not least , I would like to express my deep sense of gratitude and earnest thanks giving to my dear parents for their moral support and heartfelt cooperation in doing the main project.

Abstract

Sentiment analysis or opinion mining is the computational study of people's opinions, sentiments, attitudes, and emotions expressed in written language. It is one of the most active research areas in natural language processing and text mining in recent years. Its popularity is mainly due to two reasons. First, it has a wide range of applications because opinions are central to almost all human activities and are key influence of our behaviors. Whenever we need to make a decision, we want to hear others' opinions. Second, it presents many challenging research problems, which had never been attempted before the year 2000. Part of the reason for the lack of study before was that there was little opinionated text in digital forms. It is thus no surprise that the inception and the rapid growth of the field coincide with those of the social media on the Web. In fact, the research has also spread outside of computer science to management sciences and social sciences due to its importance to business and society as a whole.

Contents

1. Introduction

1.1 Objective

The objective is to first search for the person's tweets with the help of their twitter handler and then classify those tweets in categories of positive and negative using Natural Language Processing.

1.2 Introduction to Sentimental Analysis

Recently, Twitter has gained significant popularity among the social network services. Lots of users often use Twitter to express feelings or opinions about a variety of subjects. Analyzing this kind of content can lead to useful information for fields, such as personalized marketing or social profiling. However such a task is not trivial, because the language used in Twitter is often informal presenting new challenges to text analysis.

Sentiment Analysis is the process of determining whether a piece of writing is positive, neutral or negative. A sentiment analysis combines Natural Language Processing (NLP) and machine learning techniques to assign weighted sentiment scores to the entities, topics, themes and categories within a sentence or phrase.

Sentiment analysis helps data analysts within large enterprises gauge public opinion conduct market research, monitor brand and product reputation, and understand customer experiences in addition, data analytic companies often integrate third-party sentiment analysis APIs into their own customer experience management, social media monitoring, or workforce analytic platform, in order to deliver useful insights to their own customers.

The sentimental analysis task can be broken down into two steps

- Creating a bag of words model
- Apply classification model on bag of words model

1.3 Bag of Words Model

The bag of words model is a way of representing text data when modeling text with machine learning algorithms. The bag of words model is simple to understand and implement and has seen great success in problems such as language modeling and document classification.

The problem with Text

A problem with modeling text is that it is messy, and techniques like machine earning algorithms prefer well defined fixed length inputs and outputs. Machine learning algorithms cannot work with raw text directly. The text must be converted into numbers. Specifically, vector of numbers. This is called feature extraction, a popular simple method of feature extraction with text data is called the bag of words model of text.

What is a Bag of Words?

A bag of words model is a way of extracting features from text for use in modeling, such as with machine learning algorithms. The approach is very simple and flexible, and can be used in a number of ways for extracting features from documents.

A bag of words is a representation of text that describes the occurrence of words within a document. It involves two things:

- A vocabulary of known words.
- A measure of the presence of known words.

It is called bag of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

This model can be as simple or complex as you like. The complexity comes both in deciding how to design the vocabulary of known words and how to score the presence of known words.

Example of the Bag of Words Model

Let's make the bag of words model concrete with a worked example.

Step 1 : Collect Data

Below is a snippet of the first few lines of text from the book "Ä tale of two cities"

It was the best of times,

it was the worst of times,

it was the age of wisdom,

it was the age of foolishness,

For this example, let's treat each line as a separate document and the 4 lines as our entire corpus of documents.

Step 2 : Design the Vocabulary

Now we can make a list of all of the words in our model vocabulary

The unique words here (ignoring case and punctuation) are:

- it
- was
- the
- best
- of
- times
- worst
- age
- wisdom
- foolishness

This is a vocabulary of 10 words from a corpus containing 24 words

Step 3: Create Document Vectors

The next step is to score the words in each document.

The objective is to turn each document of free text into a vector that we can use as input or output for a machine learning model.

Because we know the vocabulary has 10 words, we can use a fixed-length document representation of 10, with one position in the vector to score each word.

The simplest scoring method is to mark the presence of words as a Boolean value, 0 for absent, 1 for present.

Using the arbitrary ordering of words listed above in our vocabulary, we can step through the first document ("*It was the best of times*") and convert it into a binary vector.

The scoring of the document would look as follows:

"it" = 1

"was" = 1

"the" = 1

"best" = 1

"of" = 1

"times" = 1

"worst" = 0

"age" = 0

"wisdom" = 0

"foolishness" = 0

As a binary vector, this would look as follows:

[1 , 1 , 1 , 1 , 1 , 1 , 0 , 0 , 0 , 0]

The other three documents would look as follows:

"it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]

"it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]

"it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

New documents that overlap with the vocabulary of known words, but may contain words outside of the vocabulary, can still be encoded , where only the occurrence of known words are scored and unknown words are ignored. You can see how this might naturally scale to large vocabularies and larger documents.

Limitations of Bag-of-Words

The bag-of-words model is very simple to understand and implement and offers a lot of flexibility for customization on your specific text data.
It has been used with great success on prediction problems like language modeling and documentation classification.
Nevertheless, it suffers from some shortcomings, such as:

- **Vocabulary**:

The vocabulary requires careful design, most specifically in order to manage the size, which impacts the sparsity of the document representations.

- **Sparsity**:

Sparse representations are harder to model both for computational reasons (space and time complexity) and also for information reasons, where the challenge is for the models to harness so little information in such a large representational space.

- **Meaning**:

Discarding word order ignores the context, and in turn meaning of words in the document (semantics). Context and meaning can offer a lot to the model, that if modeled could tell the difference between the same words differently arranged ("this is interesting" vs "is this interesting"), synonyms ("old bike" vs "used bike"), and much more.

1.4 Classification Model

1.4.1 Logistic Regression

Logistic Regression is another technique borrowed by machine learning from the field of statistics. It is the go-to method for binary classification problems(problems with two classes).

Logistic Function

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

$$1 / (1 + e^{-value})$$

Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

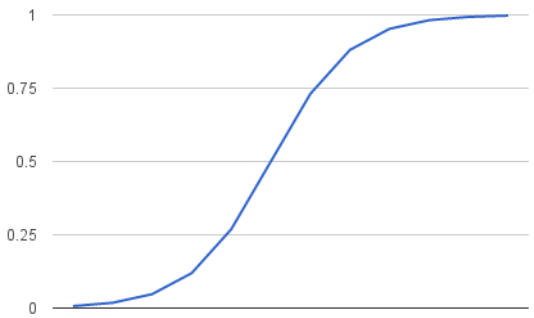


Fig.1 Logistic Function

Representation Used for Logistic Regression

Logistic regression uses an equation as the representation, very much like linear regression.

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

Eq-1 : $n(p(X) / 1 - p(X)) = b0 + b1 * X$

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

Logistic Regression Predicts Probabilities (Technical Interlude)

Logistic regression models the probability of the default class (e.g. the first class).

For example, if we are modeling people's sex as male or female from their height, then the first class could be male and the logistic regression model could be written as the probability of male given a person's height, or more formally:

P(sex=male|height)

2. Software Requirements

2.1 Anaconda

The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 15 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with scikit-learn, Tensor Flow, and Theano

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator :

- Jupyter Lab
- Jupyter Notebook
- QtConsole
- Spyder
- Glueviz
- Orange 3 App
- Rodeo
- RStudio

2.2 Spyder

Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It offers a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

Beyond its many built-in features, its abilities can be extended even further via its plug-in system and API. Furthermore, Spyder can also be used as a PyQt5 extension library, allowing developers to build upon its functionality and embed its components, such as the interactive console, in their own PyQt software.

Spyder Components

- **Editor :** Work efficiently in a multi-language editor with a function/class browser, code analysis tools, automatic code completion, horizontal/vertical splitting, and go-to-definition.
- **IPython Console :** Harness the power of as many IPython consoles as you like within the flexibility of a full GUI interface; run your code by line, cell, or file; and render plots right inline.
- **Variable Explorer :** Harness the power of as many IPython consoles as you like within the flexibility of a full GUI interface; run your code by line, cell, or file; and render plots right inline.

2.3 PyQt5

Qt is set of cross-platform C++ libraries that implement high-level APIs for accessing many aspects of modern desktop and mobile systems. These include location and positioning services, multimedia, NFC and Bluetooth connectivity, a Chromium based web browser, as well as traditional UI development.

PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative

application development language to C++ on all supported platforms including iOS and Android.

PyQt5 may also be embedded in C++ based applications to allow users of those applications to configure or enhance the functionality of those application

3. Training & Testing Data Set

3.1 Word Cloud

Positive Word Cloud

Fig.2

Negative Word Cloud

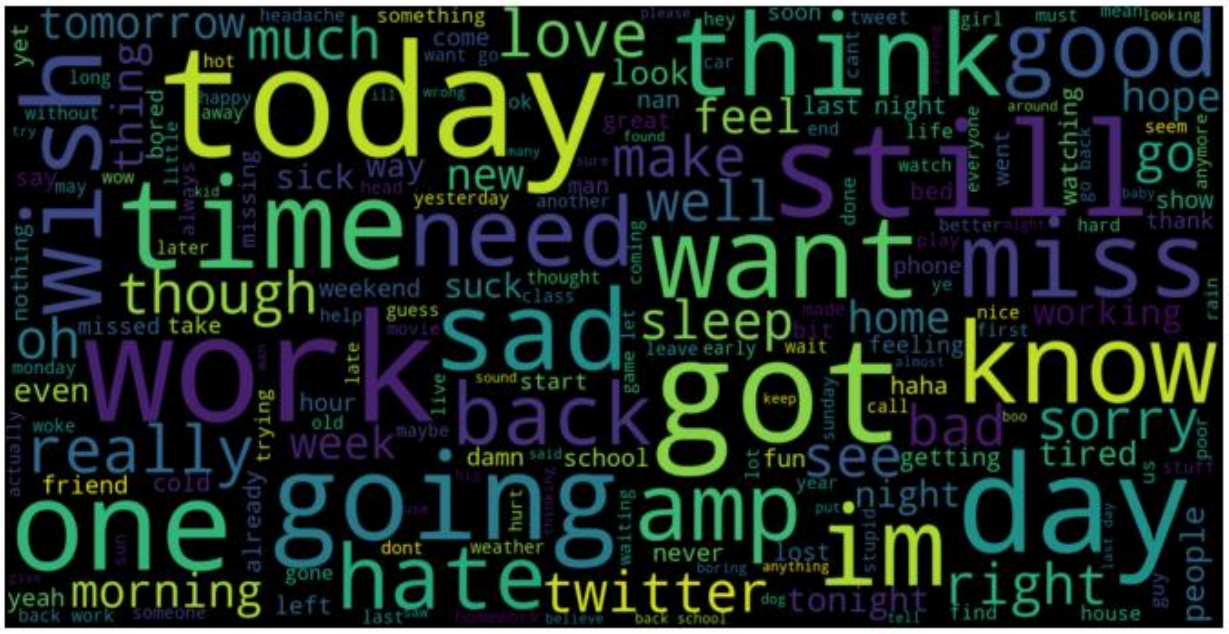


Fig.3

4. Conclusion

Nowadays, sentiment analysis or opinion mining is a hot topic in machine learning. We are still far to detect the sentiments of s corpus of texts very accurately because of the complexity in the English language and even more if we consider other languages such as Chinese.

In this project we tried to show the basic way of classifying tweets into positive or negative category using Logistic Regression as baseline and how language models are related and can produce better results. We could further improve our classifier by trying to extract more features from the tweets, trying different kinds of features, tuning the parameters of the classifier, or trying another classifier all together.

References/Bibliography

www.geeksforgeeks.com

www.numpy.org

www.matplotlib.org

www.promptcloud.com

http://www.anaconda.org

http://www.towardsdatascience.com

5. Appendices

Creating GUI

```
1.
   #importig libraries
2.import sys
3.import SearchTweets
4.import pandas as pd
5.from PyQt5.QtWidgets import *
6.from PyQt5.QtGui import *
7.from PyQt5.QtCore import *
8.
9.#main window
10.class MainWindow(QMainWindow):
11.
12.    def __init__(self, parent=None):
13.        super(MainWindow, self).__init__(parent)
14.        self.setWindowTitle("")
15.        self.form_widget = FormWidget(self)
16.        #_widget =QWidget()
17.        #_layout = QVBoxLayout(_widget)
18.        #_layout.addWidget(self.form_widget)
19.        self.setCentralWidget(self.form_widget)
20.
21.        mainMenu = self.menuBar()
22.        fileMenu = mainMenu.addMenu('File')
23.        editMenu = mainMenu.addMenu('Edit')
24.        viewMenu = mainMenu.addMenu('View')
25.        searchMenu = mainMenu.addMenu('Search')
26.        toolsMenu = mainMenu.addMenu('Tools')
27.        helpMenu = mainMenu.addMenu('Help')
28.
29.        fontButton = QAction('Font', self)
30.        fontButton.triggered.connect(self.on_click)
31.
32.        toolsMenu.addAction(fontButton)
33.        self.setFixedSize(853.33,480)
34.
35.
36.
37.
38.    @pyqtSlot()
39.    def on_click(self):
40.        print('PyQt5 button click')
41.        font, ok = QFontDialog.getFont()
42.        if ok:
43.            self.setFont(font)
44.
45.
46.class FormWidget(QWidget):
47.
48.    def __init__(self, parent):
```

```
        super(FormWidget, self).__init__(parent)
        self.__layout()
        self.__controls()


    def __layout(self):

         label = QLabel(self)
         pixmap = QPixmap('Image/thumb-1920-473471.jpg')
         pixmap = pixmap.scaled(853.3,480)
         label.setPixmap(pixmap)
         self.resize(pixmap.width(),pixmap.height())
         label =QLabel(self)
         label.setText("Enter Name to search Tweets")
         label.setStyleSheet('color:blue')
         label.setFont(QFont('MSShellDlg2',14))
         label.resize(250,50)
         label.move(50,50)
         self.setFont(QFont('MSShellDlg2',14))
         #text box to get screenName
         self.textbox = QLineEdit(self)
         self.textbox.move(50, 100)
         self.textbox.resize(250,25)
         self.textbox.setStyleSheet('color:blue')
         srchButton = QPushButton('SEARCH', self)
         srchButton.setToolTip('search tweets')
         srchButton.move(50,150)
         srchButton.clicked.connect(self.search)
         #self.createTable()
         '''''self.layout = QVBoxLayout()
         self.layout.addWidget(self.tableWidget)
         self.setLayout(self.layout)'''

    def __controls(self):
         print("",end="")


    @pyqtSlot()
    def search(self):
        global screenName
        screenName = self.textbox.text()
        #tweets = SearchTweets.get_all_tweets(self.textbox.text())
        self.table=Table(self)
        self.table.show()
             #print(tweets)
             #self.createTable(len(tweets),2)


class Table(QWidget):

    def __init__(self, parent=None):
             super(Table, self).__init__()
             self.__layout()
             self.__controls()

             self.setFixedSize(853.33,480)
         def __layout(self):
             label =QLabel(self)
```

```
            label.setText("Enter Name to search Tweets")
            label.setStyleSheet('color:blue')
            label.setFont(QFont('MSShellDlg2',14))
            label.resize(250,50)
            label.move(50,50)
            self.createTable()
            self.layout = QVBoxLayout()
            self.layout.addWidget(self.tableWidget)
            self.setLayout(self.layout)


        def __controls(self):
            print("",end="")

        def createTable(self):
            #get all tweets along with prediction
            dataset =  SearchTweets.get_all_tweets(screenName)
            self.tableWidget = QTableWidget()
            self.tableWidget.setVerticalHeaderLabels(('Id', 'Created_At', 'Tweet','Status'))
            self.tableWidget.setRowCount(len(dataset))
            self.tableWidget.setColumnCount(4)
            self.tableWidget.setColumnWidth(0, 30)
            self.tableWidget.setColumnWidth(1, 160)
            print(len(dataset))

            for r in range(0,len(dataset)):

                self.tableWidget.setItem(r, 0, QTableWidgetItem(str(dataset.iat[r,0])))
                self.tableWidget.setItem(r, 1, QTableWidgetItem(dataset.iat[r,1]))
                self.tableWidget.setItem(r, 2, QTableWidgetItem(dataset.iat[r,2]))
                self.tableWidget.setItem(r, 3, QTableWidgetItem(dataset.iat[r,3]))
            self.tableWidget.move(0, 0)


    def main():
        app = QApplication(sys.argv)
        win = MainWindow()
        win.show()
        app.exec_()

    if __name__ == '__main__':
        sys.exit(main())
```

Searching Tweets

```
import pandas as pd
import tweepy
import classifyTweets
import os
import tweepy
import csv
```

```
import json

# load Twitter API credentials

with open('twitter_credentials.json') as cred_data:
    info = json.load(cred_data)
    consumer_key = info['CONSUMER_KEY']
    consumer_secret = info['CONSUMER_SECRET']
    access_key = info['ACCESS_KEY']
    access_secret = info['ACCESS_SECRET']

def get_all_tweets(screen_name):

    # Authorization and initialization
    if(not os.path.exists(screen_name+'_tweets.csv')):
        auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
        auth.set_access_token(access_key, access_secret)
        api = tweepy.API(auth)

        # initialization of a list to hold all Tweets

        all_the_tweets = []

        # We will get the tweets with multiple requests of 200 tweets e
    ach

        new_tweets = api.user_timeline(screen_name=screen_name, count=2
    00)

        # saving the most recent tweets

        all_the_tweets.extend(new_tweets)

        # save id of 1 less than the oldest tweet

        oldest_tweet = all_the_tweets[-1].id - 1

        # grabbing tweets till none are left

        while (len(new_tweets) !=  0):
            # The max_id param will be used subsequently to prevent dup
    licates
            new_tweets = api.user_timeline(screen_name=screen_name,
            count=200, max_id=oldest_tweet)

            # save most recent tweets

            all_the_tweets.extend(new_tweets)

            # id is updated to oldest tweet - 1 to keep track

            oldest_tweet = all_the_tweets[-1].id - 1
            print ('...%s tweets have been downloaded so far' % len(all
    _the_tweets))

        # transforming the tweets into a 2D array that will be used to
    populate the csv

```

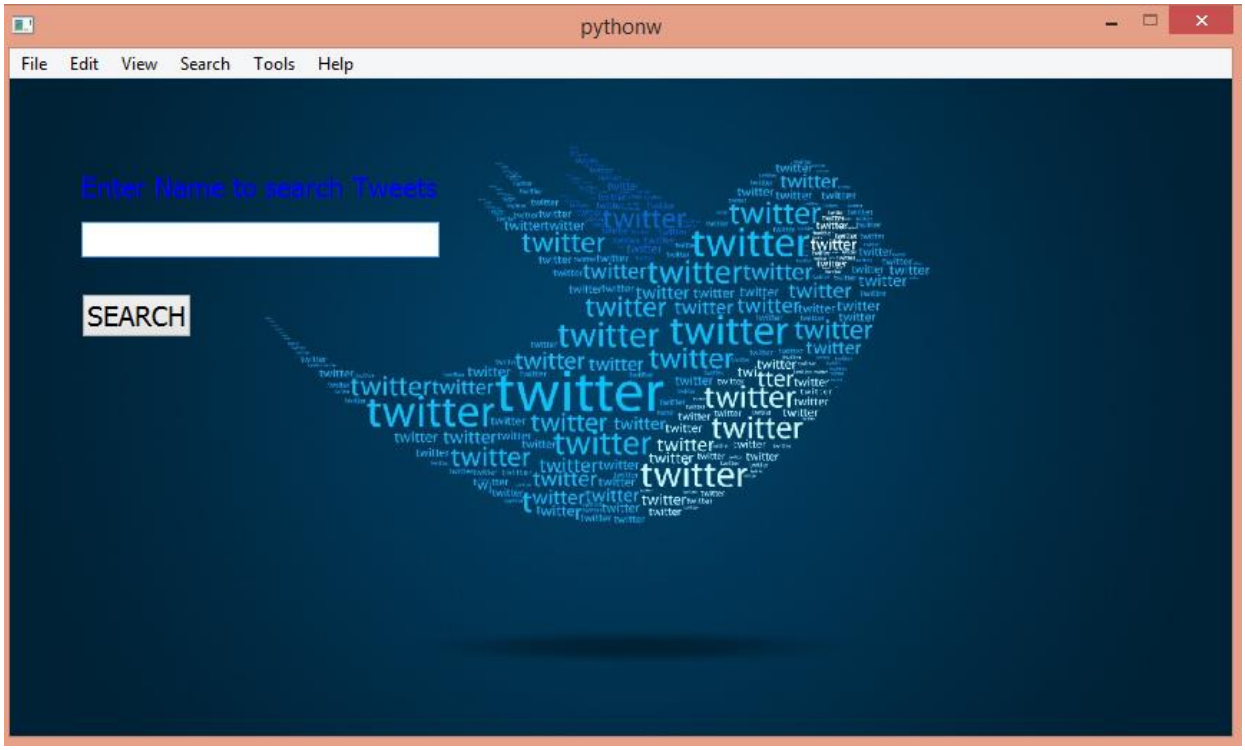
```
        outtweets = [[tweet.id_str, tweet.created_at,
        tweet.text.encode('utf-8')] for tweet in all_the_tweets]

        # writing to the csv file

        with open(screen_name + '_tweets.csv', 'w', encoding='utf8') as
    f:
            writer = csv.writer(f)
            writer.writerow(['id', 'created_at', 'text'])
            writer.writerows(outtweets)

    # passign tweets to get prediction
    return classifyTweets.getPredictedResult(screen_name)
```

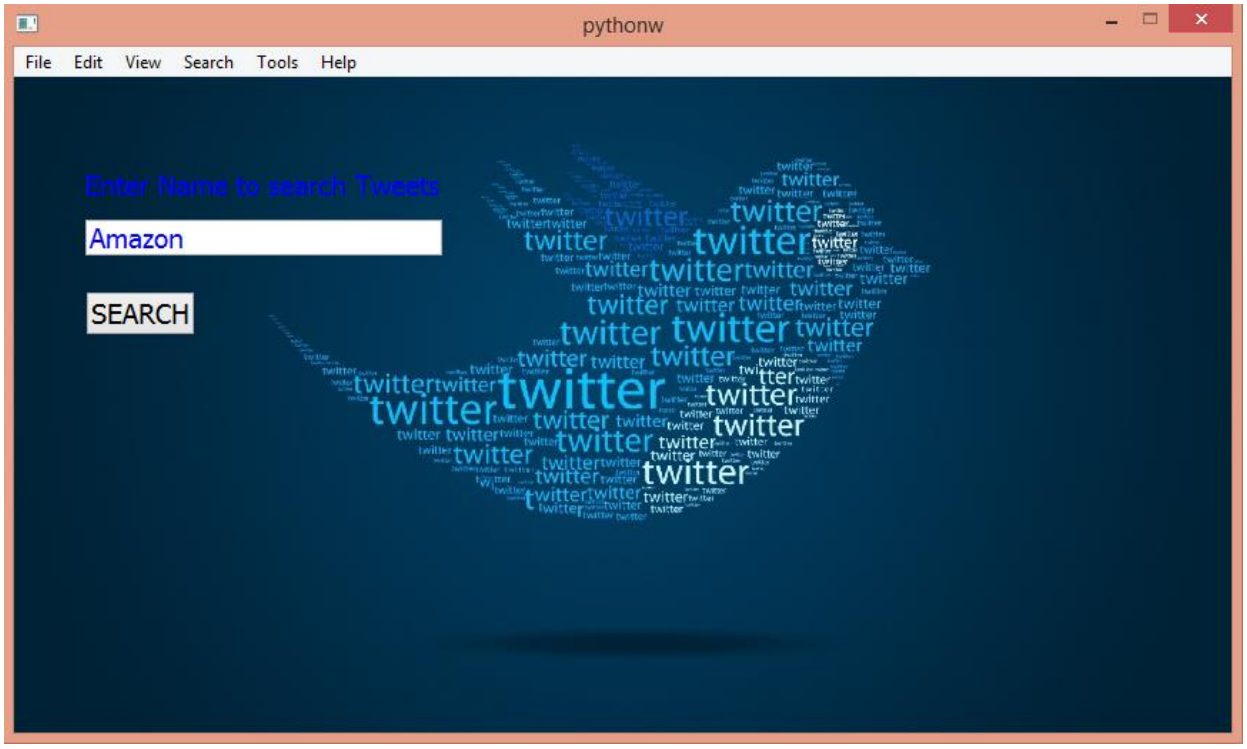
Getting Predicted Data

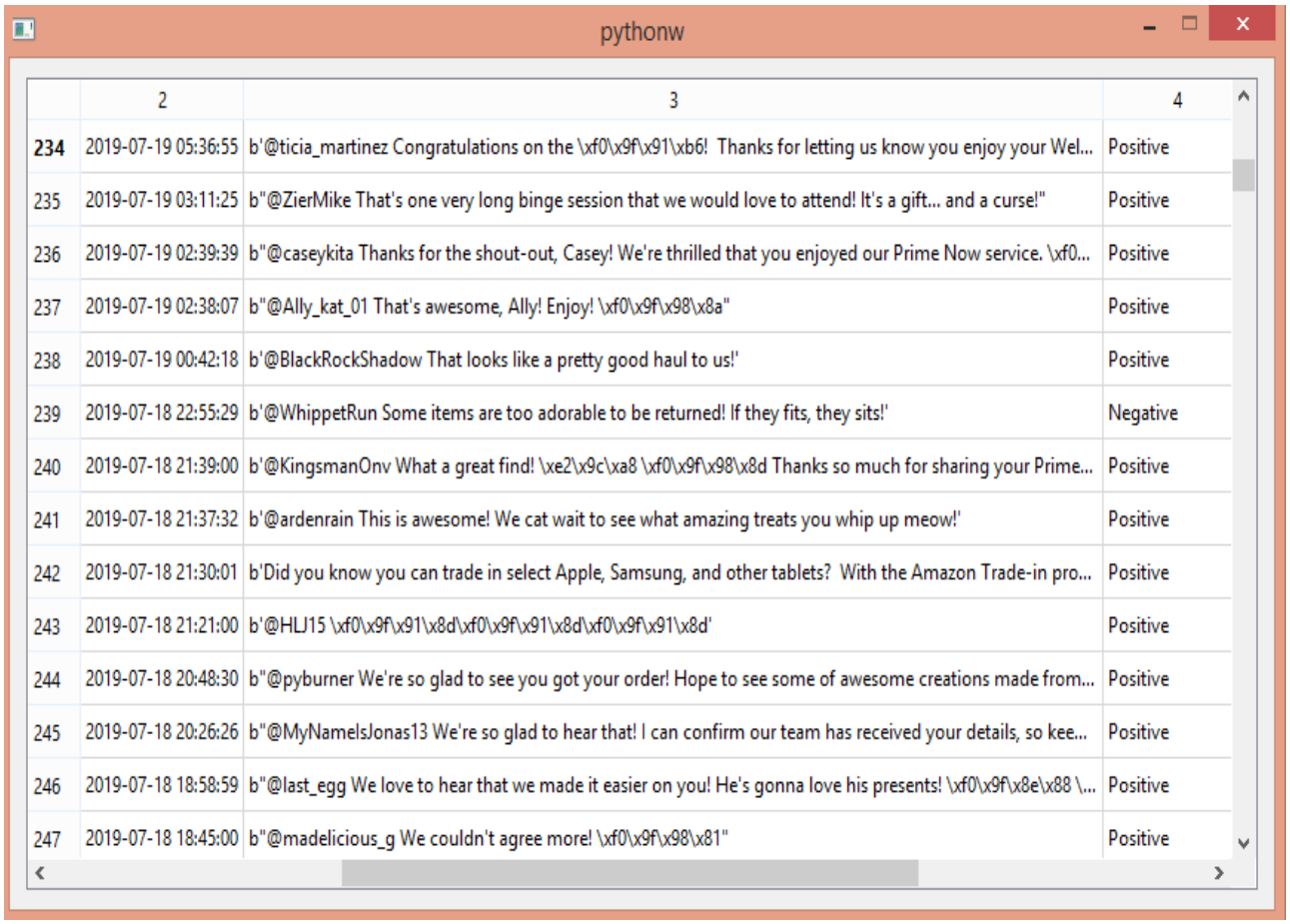
```
# importing libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import re
import nltk
import os.path
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer

# getting common words dictionary
with open('words2.txt') as word_file:
  words = set(word_file.read().split())

# getting training dataset
dataset = pd.read_csv('trainingData.csv',encoding='iso-8859-
1',names=['Status','Tweets'])

# training the model
def trainModel():
    global refTweets
    global classifier
    global cv
    if(not os.path.exists('refinedTweets.csv')):
        # cleaning data
        for i in range(0, len(dataset)):
            tweet = re.sub('[^a-zA-Z]', ' ', dataset['Tweet'][i])
            tweet = tweet.lower()
            tweet = tweet.split()
            ps = PorterStemmer()
            tweet = [ps.stem(word) for word intweet if not word in set(
stopwords.words('english'))]
            tweet= ' '.join(tweet)
            refTweets.append(tweet)
    else:
         refTweets = pd.read_csv('refinedTweets.csv')

    A = pd.DataFrame({'RefTweets':refTweets.iloc[0:50000,0].values})
    A = A['RefTweets'].astype(str).tolist()
```

```
    B = pd.DataFrame({'RefTweets':refTweets.iloc[998576:,0].values})
    B = B['RefTweets'].astype(str).tolist()
    A.extend(B)
    refTweets = A

    # preparing dataset
    cv = CountVectorizer(max_features = 3000)
    X = cv.fit_transform(refTweets).toarray()
    y1 = dataset.iloc[0:50000, 0].values
    y2 = dataset.iloc[998576:, 0].values
    testY= np.concatenate((y1,y2))
    y=testY
    # Fitting Logistic Regression to the Training set
    from sklearn.linear_model import LogisticRegression
    classifier = LogisticRegression(random_state = 0)
    classifier.fit(X, y)
    print("Model traied")


def getPredictedResult(screen_name):
      global tweets
      global corpus
      global predictedY
      tweets = pd.read_csv(screen_name+'_tweets.csv')
      corpus=[]
      print(len(tweets))
      # cleaning the data
      for i in range(0, len(tweets)):
            tweet = re.sub('[^a-zA-Z]', ' ', tweets['text'][i])
            tweet = tweet.lower()
            tweet = tweet.split()
            ps = PorterStemmer()
            tweet = [ps.stem(word) for word in tweet if not word in set
  (stopwords.words('english'))]
            tweet = [word for word in tweet if word in words]
            tweet = ' '.join(tweet)
            corpus.append(tweet)
      X = cv.transform(corpus).toarray()
      # predicting the result
      predictedY = classifier.predict(X)
      Status=[]
      for i in predictedY :
        if(i==0):
          Status.append('Negative')
        else:
          Status.append('Positive')
      tweets['Status'] = Status

      return tweets
# when first time script will be loaded model will be trained
trainModel()
```

Outcome :


pythonw
File Edit View Search Tools Help
Enter Name to search Tweets
SEARCH


pythonw
File Edit View Search Tools Help
Enter Name to search Tweets
Amazon
SEARCH



	2	3	4
234	2019-07-19 05:36:55	b'@ticia_martinez Congratulations on the \xf0\x9f\x91\xb6! Thanks for letting us know you enjoy your Wel...	Positive
235	2019-07-19 03:11:25	b"@ZierMike That's one very long binge session that we would love to attend! It's a gift... and a curse!"	Positive
236	2019-07-19 02:39:39	b"@caseykita Thanks for the shout-out, Casey! We're thrilled that you enjoyed our Prime Now service. \xf0...	Positive
237	2019-07-19 02:38:07	b"@Ally_kat_01 That's awesome, Ally! Enjoy! \xf0\x9f\x98\x8a"	Positive
238	2019-07-19 00:42:18	b'@BlackRockShadow That looks like a pretty good haul to us!'	Positive
239	2019-07-18 22:55:29	b'@WhippetRun Some items are too adorable to be returned! If they fits, they sits!'	Negative
240	2019-07-18 21:39:00	b'@KingsmanOnv What a great find! \xe2\x9c\xa8 \xf0\x9f\x98\x8d Thanks so much for sharing your Prime...	Positive
241	2019-07-18 21:37:32	b'@ardenrain This is awesome! We cat wait to see what amazing treats you whip up meow!'	Positive
242	2019-07-18 21:30:01	b'Did you know you can trade in select Apple, Samsung, and other tablets? With the Amazon Trade-in pro...	Positive
243	2019-07-18 21:21:00	b'@HLJ15 \xf0\x9f\x91\x8d\xf0\x9f\x91\x8d\xf0\x9f\x91\x8d'	Positive
244	2019-07-18 20:48:30	b"@pyburner We're so glad to see you got your order! Hope to see some of awesome creations made from...	Positive
245	2019-07-18 20:26:26	b"@MyNamelsJonas13 We're so glad to hear that! I can confirm our team has received your details, so kee...	Positive
246	2019-07-18 18:58:59	b"@last_egg We love to hear that we made it easier on you! He's gonna love his presents! \xf0\x9f\x8e\x88 \...	Positive
247	2019-07-18 18:45:00	b"@madelicious_g We couldn't agree more! \xf0\x9f\x98\x81"	Positive