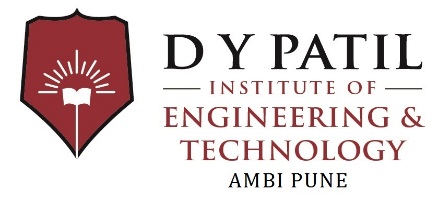
****

**“Twitter Sentiment Analysis In Python”**

Submitted in partial fulfillment of

The requirement for the

**DEGREE OF ENGINEERING**

**In**

**COMPUTER ENGINEERING**

Department of Computer Engineering

Dr. D. Y. Patil Institute of Engineering and Technology,

Ambi, Pune

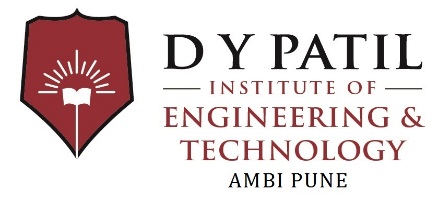
Year: 2020-21

Semester: V

**D. Y. Patil Institute of Engineering and Technology**

**Ambi, Pune**

**Department of computer Engineering**

****

**CERTIFICATE**

This is certify that

**Karthik Arumugam & Yogesh Rajgure**

have completed the Mini Project entitled

**“Twitter Sentiment Analysis In Python”**

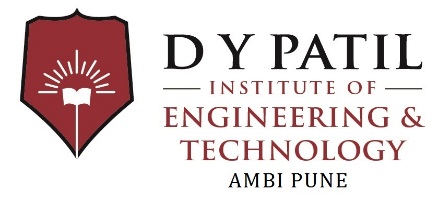
Satisfactorily for the partial fulfillment of the requirement for the

Degree in Computer Engineering from

University of Pune during Academic Year 2020-21

(Prof. Rohini Hanchate) (Prof. Mangesh Manke)

Report Guide HOD Computer Dept.

****

**Report on**

**“Twitter Sentiment Analysis In Python”**

Submitted in partial fulfillment of

The requirement for

**DEGREE OF ENGINEERING**

**In**

**COMPUTER ENGINEERING**

Submitted By:

Karthik Arumugam -Roll no: 7

Yogesh Rajgure - Roll no: 20

Batch: 2020-21

Department of Computer Engineering

Dr. D. Y. Patil Institute of Engineering and Technology,

Ambi, Pune

Year: 2020-21

Semester: I

**Acknowledgement**

Being very much motivated towards in the field python that deals with collecting data and bringing useful insights for the same, we would like to thank Rohini Hanchate ma’am for giving this wonderful opportunity to work on a project like this.

Karthik Arumugam Roll no: 7

Yogesh Rajgure Roll no: 20

T.E. Computer (I) 2020-21

Batch: 2020-21

**Abstract**

This project addresses the problem of sentiment analysis in twitter; that is classifying tweets according to the sentiment expressed in them: positive, negative or neutral. Twitter is an online micro-blogging and social-networking platform which allows users to write short status updates of maximum length 140 characters. It is a rapidly expanding service with over 200 million registered users, out of which 100 million are active users and half of them log on twitter on a daily basis - generating nearly 250 million tweets per day. Due to this large amount of usage we hope to achieve a reflection of public sentiment by analyzing the sentiments expressed in the tweets. Analyzing the public sentiment is important for many applications such as firms trying to find out the response of their products in the market, predicting political elections and predicting socioeconomic phenomena like stock exchange. The aim of this project is to develop a functional classifier for accurate and automatic sentiment classification of an unknown tweet stream.

**Contents**

|  |  |  |
| --- | --- | --- |
| **Sr.no** | **Contents** | **Pg. No** |
|  |  |  |
| 1. | Introduction | 7 |
| 2. | Mini Project Details | 8 |
| 3. | Software & Hardware requirements | 10 |
| 4. | Implementation | 11 |
| 5 | Source Code | 14 |
| 6. | Screenshot | 18 |
| 7. | Conclusion | 20 |

**INTRODUCTION**

We have chosen to work with twitter since we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for twitter, as compared to traditional blogging sites. More over the response on twitter is more prompt and also more general (since the number of users who tweet is substantially more than those who write web blogs on a daily basis). Sentiment analysis of public is highly critical in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm. This could be done by analyzing overall public sentiment towards that firm with respect to time and using economics tools for finding the correlation between public sentiment and the firm’s stock market value. Firms can also estimate how well their product is responding in the market, which areas of the market is it having a favorable response and in which a negative response since twitter allows us to download stream of geo-tagged tweets for particular locations. If firms can get this information, they can analyze the reasons behind geographically differentiated response, and so they can market their product in a more optimized manner by looking for appropriate solutions like creating suitable market segments. Predicting the results of popular political elections and polls is also an emerging application to sentiment analysis. One such study was conducted by Tumasjan et al. in Germany for predicting the outcome of federal elections in which concluded that twitter is a good reflection of offline sentiment.

**Project Introduction: -**

This project of analyzing sentiments of tweets comes under the domain of “Pattern Classification” and “Data Mining”. Both of these terms are very closely related and intertwined, and they can be formally defined as the process of discovering “useful” patterns in large set of data, either automatically (unsupervised) or semiautomatically (supervised).

The project would heavily rely on techniques of “Natural Language Processing” in extracting significant patterns and features from the large data set of tweets and on “Machine Learning” techniques for accurately classifying individual unlabeled data samples (tweets) according to whichever pattern model best describes them.

The features that can be used for modeling patterns and classification can be divided into two main groups: formal language based and informal blogging based. Language based features are those that deal with formal linguistics and include prior sentiment polarity of individual words and phrases, and parts of speech tagging of the sentence. Prior sentiment polarity means that some words and phrases have a natural innate tendency for expressing particular and specific sentiments in general. For example, the word “excellent” has a strong positive connotation while the word “evil” possesses a strong negative connotation. So whenever a word with positive connotation is used in a sentence, chances are that the entire sentence would be expressing a positive sentiment. Parts of Speech tagging, on the other hand, is a syntactical approach to the problem. It means to automatically identify which part of speech each individual word of a sentence belongs to: noun, pronoun, adverb, adjective, verb, interjection, etc. Patterns can be extracted from analyzing the frequency distribution of these parts of speech (ether individually or collectively with some other part of speech) in a particular class of labeled tweets. Twitter based features are more informal and relate with how people express themselves on online social platforms and compress their sentiments in the limited space of 140 characters offered by twitter. They include twitter hashtags, retweets, word capitalization, word lengthening, question marks, presence of URL in tweets, exclamation marks, internet emoticons and internet shorthand/slangs.

Classification techniques can also be divided into two categories:

Supervised vs. unsupervised and non-adaptive vs. adaptive/reinforcement techniques. Supervised approach is when we have pre-labeled data samples available and we use them to train our classifier. Training the classifier means to use the pre-labeled to extract features that best model the patterns and differences between each of the individual classes, and then classifying an unlabeled data sample according to whichever pattern best describes it. For example, if we come up with a highly simplified model that neutral tweets contain 0.3 exclamation marks per tweet on average while sentiment-bearing tweets contain 0.8, and if the tweet we have to classify does contain 1 exclamation mark, then (ignoring all other possible features) the tweet would be classified as subjective, since 1 exclamation mark is closer to the model of 0.8 exclamation marks. Unsupervised classification is when we do not have any labeled data for training. In addition to this adaptive classification techniques deal with feedback from the environment. In our case feedback from the environment can be in form of a human telling the classifier whether it has done a good or poor job in classifying a particular tweet and the classifier needs to learn from this feedback. There are two further types of adaptive techniques: Passive and active. Passive techniques are the ones which use the feedback only to learn about the environment (in this case this could mean improving our models for tweets belonging to each of the three classes) but not using this improved learning in our current classification algorithm, while the active approach continuously keeps changing its classification algorithm according to what it learns at real-time.

Mini Project Details: -

Sentiment analysis, also refers as opinion mining, is a sub machine learning task where we want to determine which is the general sentiment of a given document. Using machine learning techniques and natural language processing we can extract the subjective information of a document and try to classify it according to its polarity such as positive, neutral or negative. It is a really useful analysis since we could possibly determine the overall opinion about a selling object, or predict stock markets for a given company like, if most people think positive about it, possibly its stock markets will increase, and so on.

Sentiment analysis is actually far from to be solved since the language is very complex (objectivity/subjectivity, negation, vocabulary, grammar, …) but it is also why it is very interesting to working on.

In this project we choose to try to classify tweets from Twitter into “positive”, “negative” and “neutral” sentiment by building a model based on probabilities. Twitter is a microblogging website where people can share their feelings quickly and spontaneously by sending a tweets limited by 140 characters. You can directly address a tweet to someone by adding the target sign “@” or participate to a topic by adding a hashtag “#” to your tweet. Because of the usage of Twitter, it is a perfect source of data to determine the current overall opinion about anything.

After classifying the tweets based on emotions, we have plotted the data on a pie chart using matplotlib to have a better view at how the data is distributed.

Software & Hardware Requirements: -

Software: -

1. Windows Operating system (8/10)/Linux OS
2. Python interpreter 3.8

Hardware: -

1. Intel Core processor
2. Ram 2GB or above

Implementation: -

The whole program is written in python language.

So, we import the necessary modules.

We first begin by creating a SentimentTwitterProject class. This class contains all the methods to interact with Twitter API and parsing tweets. We use **\_\_init\_\_**function to handle the authentication of API client.

In **get\_tweets**function, we use: -

fetched\_tweets = self.api.search(q = query, count = count) to call the twitter API to fetch tweets.

In **get\_tweet\_sentiment**we use textblob module: -

analysis = TextBlob(self.clean\_tweet(tweet))

TextBlob is actually a high-level library built over top of Natural Language toolkit library. First, we call **clean\_tweet** method to remove links, special characters, etc. from the tweet using some simple regex.  
Then, as we pass **tweet** to create a **TextBlob** object, following processing is done over text by textblob library:

* Tokenize the tweet i.e., split words from body of text.
* Remove stopwords from the tokens. (stopwords are the commonly used words which are irrelevant in text analysis like I, am, you, are, etc.)
* Do POS (part of speech) tagging of the tokens and select only significant features/tokens like adjectives, adverbs, etc.
* Pass the tokens to a **sentiment classifier**which classifies the tweet sentiment as positive, negative or neutral by assigning it a polarity between -1.0 to 1.0.

Below I have shown how sentiment classifier is created: -

* **TextBlob** uses a Movies Reviews dataset in which reviews have already been labelled as positive or negative.
* Positive and negative features are extracted from each positive and negative review respectively.
* Training data now consists of labelled positive and negative features. This data is trained on a Naives Bayes Classifier.

Then, we use **sentiment.polarity** method of **TextBlob** class to get the polarity of tweet between -1 to 1.

Then, we classify polarity as Positive, negative or neutral.

Finally, parsed tweets are returned so that, we can do various type of statistical analysis on the tweets.

In this program, we tried to find the percentage of positive, negative and neutral tweets about a query.

Finally, after classifying them as positive, negative or neutral along with their percentages we have then plotted them on a pie-chart to have a proper visual understanding.

Source Code :-

import re

import tweepy

from tweepy import OAuthHandler

from textblob import TextBlob

from matplotlib import pyplot as plt

class TwitterClient(object):

def \_\_init\_\_(self):

consumer\_key = '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

consumer\_secret = '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

access\_token = ‘\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

access\_token\_secret = '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

try:

self.auth = OAuthHandler(consumer\_key, consumer\_secret)

self.auth.set\_access\_token(access\_token, access\_token\_secret)

self.api = tweepy.API(self.auth)

except:

print("Error: Authentication Failed")

def clean\_tweet(self, tweet):

return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())

def get\_tweet\_sentiment(self, tweet):

analysis = TextBlob(self.clean\_tweet(tweet))

if analysis.sentiment.polarity > 0:

return 'positive'

elif analysis.sentiment.polarity == 0:

return 'neutral'

else:

return 'negative'

def get\_tweets(self, query, count = 10):

tweets = []

try:

fetched\_tweets = self.api.search(q = query, count = count)

for tweet in fetched\_tweets:

parsed\_tweet = {}

parsed\_tweet['text'] = tweet.text

parsed\_tweet['sentiment'] = self.get\_tweet\_sentiment(tweet.text)

if tweet.retweet\_count > 0:

if parsed\_tweet not in tweets:

tweets.append(parsed\_tweet)

else:

tweets.append(parsed\_tweet)

return tweets

except tweepy.TweepError as e:

print("Error : " + str(e))

def main():

api = TwitterClient()

tweets = api.get\_tweets(query = 'Coronavirus', count = 200)

ptweets = [tweet for tweet in tweets if tweet['sentiment'] == 'positive']

print("Positive tweets percentage: {} %".format(100\*len(ptweets)/len(tweets)))

ntweets = [tweet for tweet in tweets if tweet['sentiment'] == 'negative']

print("Negative tweets percentage: {} %".format(100\*len(ntweets)/len(tweets)))

print("Neutral tweets percentage: {} % ".format(100\*(len(tweets) -(len( ntweets )+len( ptweets)))/len(tweets)))

good\_t = 100\*len(ptweets)/len(tweets)

neg\_t = 100\*len(ntweets)/len(tweets)

neutral\_t = 100\*(len(tweets) -(len( ntweets )+len( ptweets)))/len(tweets)

parameter = ['Positive','Negative','Neutral']

data\_val = [good\_t,neg\_t,neutral\_t]

fig = plt.figure(figsize=(10,7))

plt.pie(data\_val,labels = parameter)

plt.show()

print("\n\nPositive tweets:")

for tweet in ptweets[:10]:

print(tweet['text'])

print("\n\nNegative tweets:")

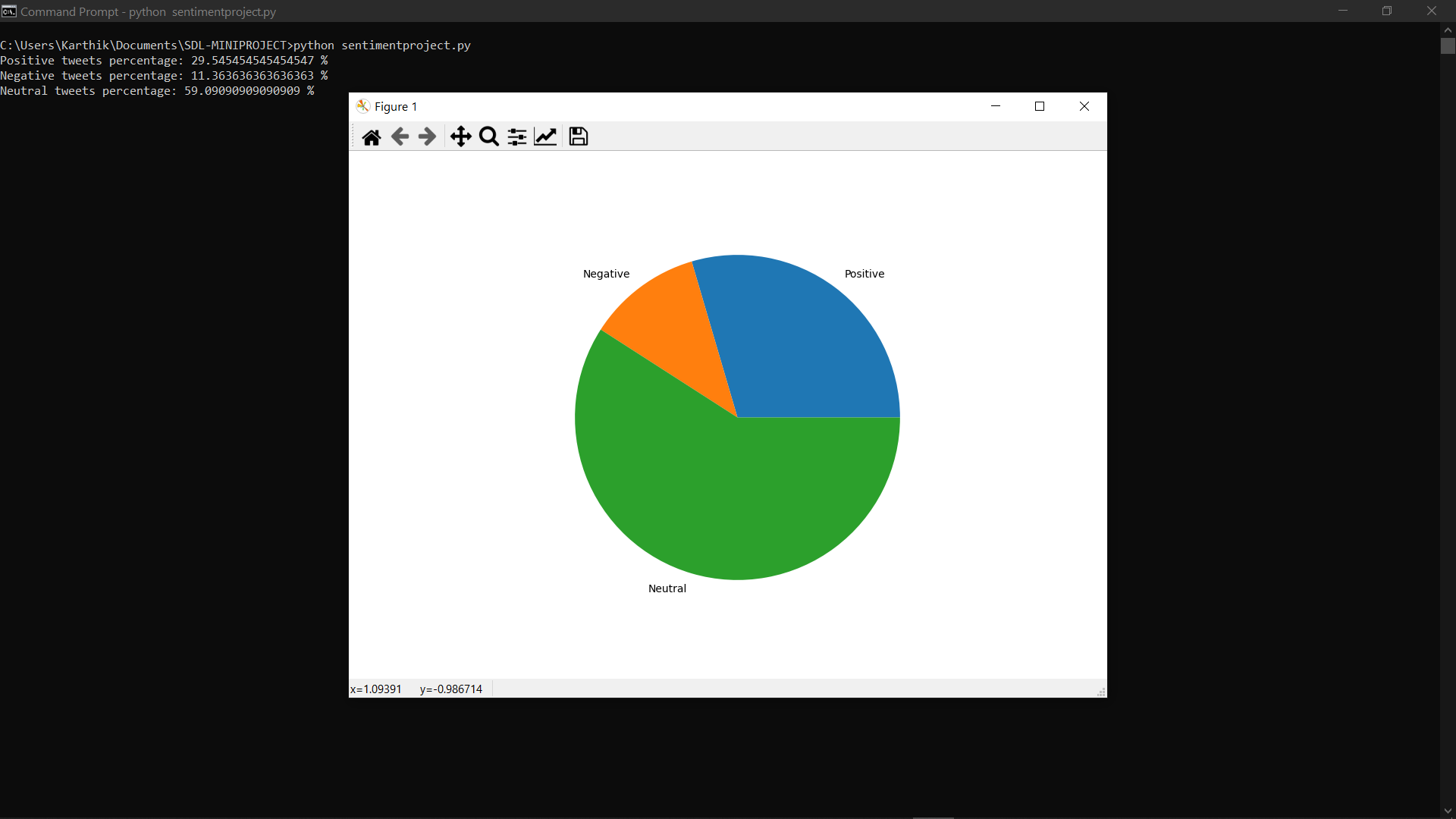
for tweet in ntweets[:10]:

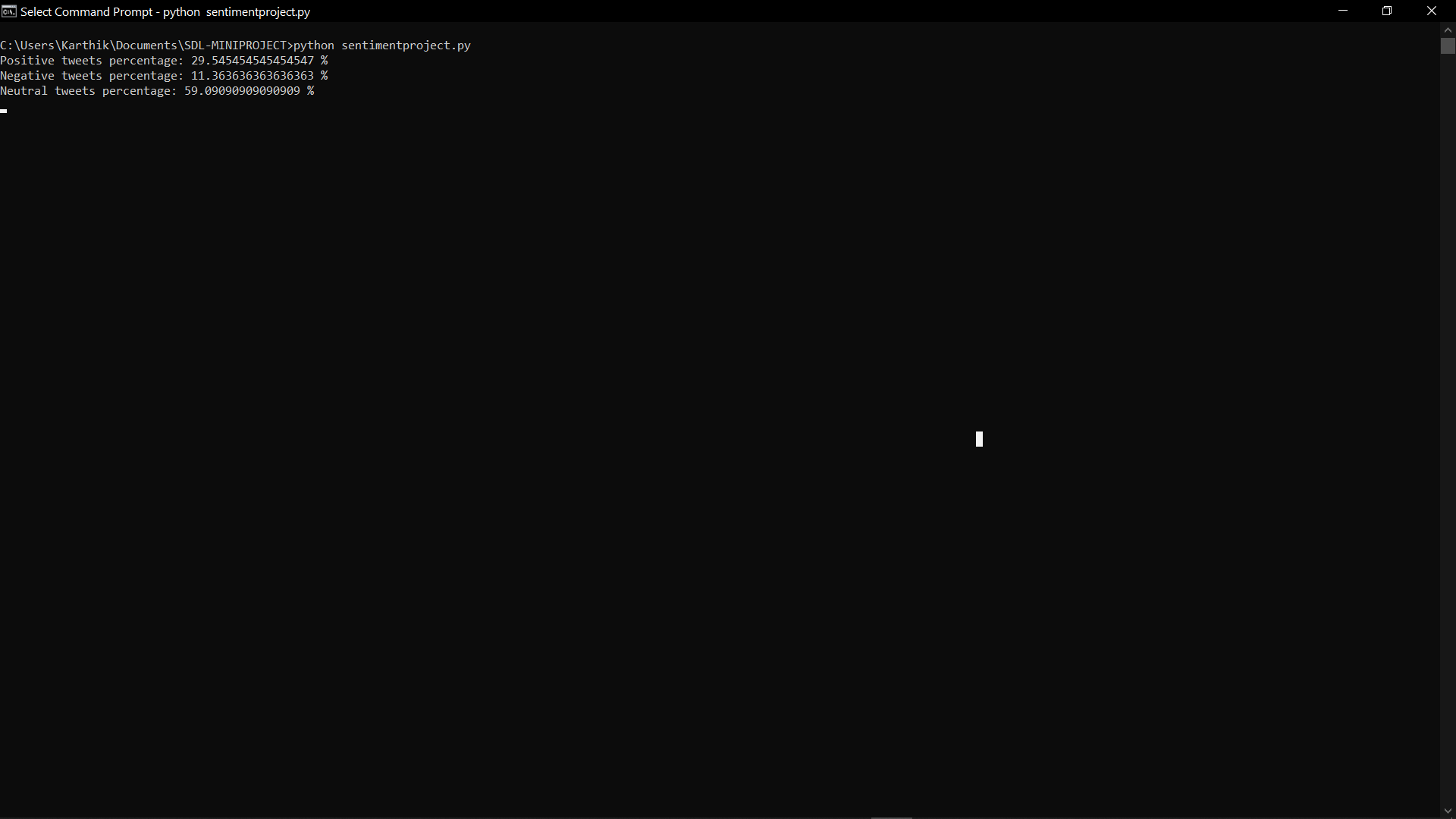
print(tweet['text'])

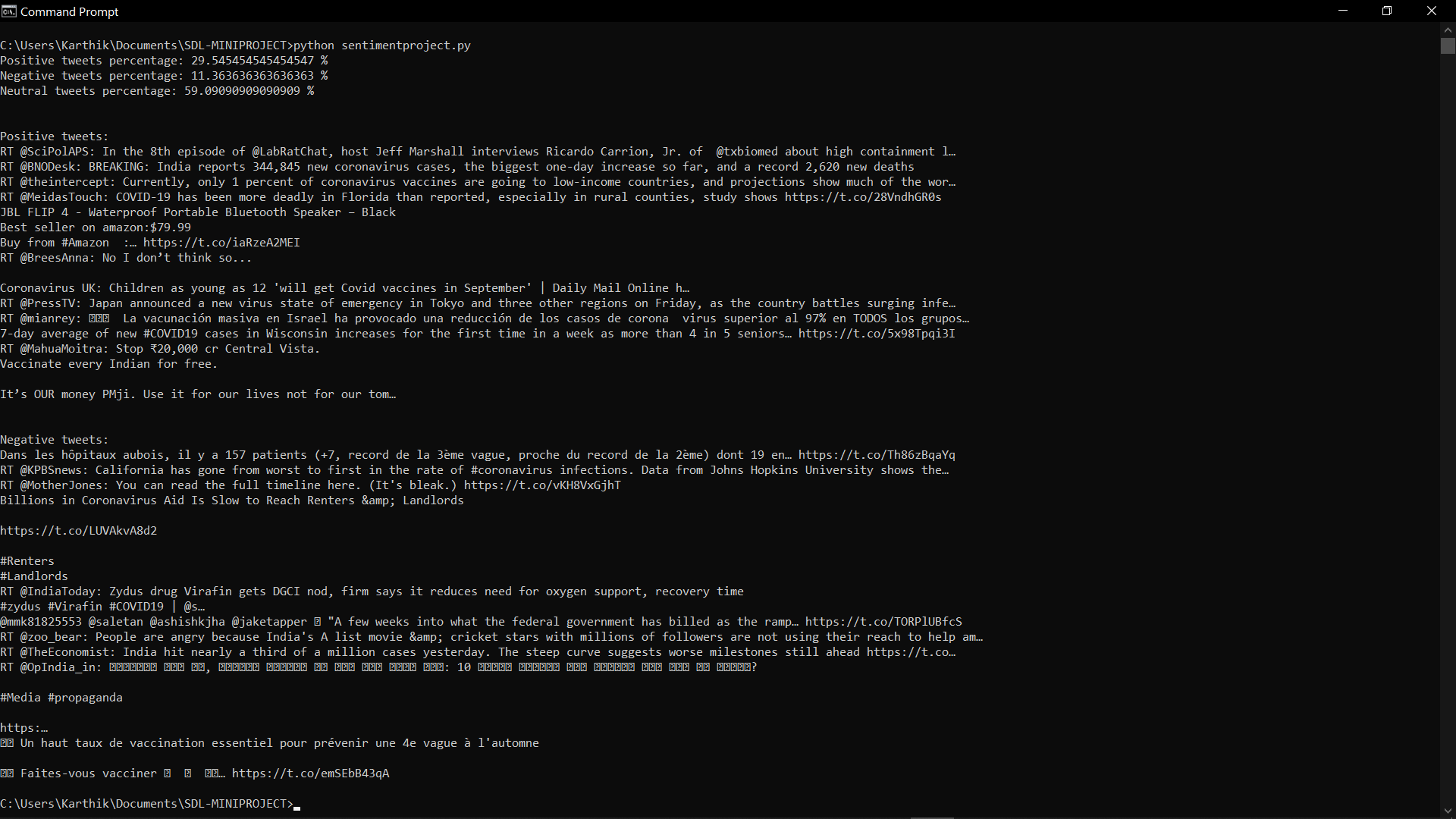
if \_\_name\_\_ == "\_\_main\_\_":

main()

Screenshots:-







Conclusion: -

Sentiment analysis is a very important topic in terms of understanding the emotions of people being expressed on any given topic. With advancements in the technology and new ways to analyze the data, sentiment analysis has become an important field in areas of machine learning as well.

Complexities such as to detect the sentiments of scorpus of texts very accurately because of the complexity in the English language and even more if we consider other languages such as Chinese need to be solved.

In this project we tried to show the basic way of classifying tweets into positive or negative category using python language. Based on my other researches the program could be further improved by trying to extract more features from the tweets, trying different kinds of features, tuning the parameters of the naïve Bayes classifier, or trying another classifier all together to make it more accurate and effective.