A

15CS375L-MINOR PROJECT REPORT

on

SENTIMENT ANAYSIS OF TWITTER DATA

Submitted in partial fulfillment of

Degree of Bachelor of Technology in Computer Science & Technology

Submitted By

LOKESH K	**Reg.No: RA1711003040036**
MAGESHWAR.S	**Reg.No: RA1711003040087**
NISHANTH.G	**Reg.No: RA1711003040113**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

VADAPALANI, CITY CAMPUS, CHENNAI

OCTOBER 2019

BONAFIDE CERTIFICATE

Certified that this project report titled **"SENTIMENT ANALYSIS OF TWITTER DATA"** is the bonafide work of **"LOKESH.K (RA1711003040036), MAGESHWAR.S (RA1711003040087), NISHANTH G (RA1711003040113)"** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE OF THE GUIDE	**SIGNATURE OF THE HOD**
Mr. D. Manikkannan, M.E.,(Ph.D)	Dr.S.Prasanna Devi, B.E.,M.E., Ph.D., PGDHRM.,PDF(IISc)
Assistant Professor	Professor & Head
Department of Computer Science and Engineering	Department of Computer Science and Engineering
SRM Institute of Science & Technology	SRM Institute of Science & Technology
Vadapalani Campus	Vadapalani Campus

ACKNOWLEDGEMENT

I place my deep sense of gratitude to our beloved Chancellor, Dr. T.R.Pachamuthu, SRM IST, for providing us with the requisite infrastructure throughout the course.

I take the opportunity to extend my heartfelt thanks for the support of our Respected Dean Dr.K.Duraivelu B.E., M.B.A., M.E., Ph.D., for his impeccable guidance.

I would like to thank the Head of Department of CSE, SRM Institute of Science and Technology, Vadapalani, Dr. Prasanna Devi B.E., M.E., Ph.D., PGDHRM. PDF (IISc), for the constant encouragement and support she gave to take up innovative projects.

I wish to extend my gratitude to Mr. D.Manikkannan, M.E., (Ph.D) Asst. Professor, CSE, for his constant advice on the various project-related issues. Without his help, it would have been really harder for us to accomplish our ideas and finish the project.

TABLE OF CONTENTS

ABSTRACT

In this current era, social media plays a important role in data exchange, sharing their thoughts. Emotional Effect of a person maintains an important role on their day to day life.

Sentiment Analysis is a procedure of analyzing the opinions and polarity of thoughts of the person. Twitter is a main platform on sharing the thought's, opinion and sentiments on different occasions. Twitter Sentimental Analysis is method of analyzing the emotions from tweets (message posted by user in twitter). Tweets are helpful in extracting the Sentimental values from the user. The data provide the Polarity indication like positive, negative or unbiased values. It is focused on the person's tweets and the hash tags for understanding the situations in each aspect of the criteria. Sentiment Analysis is the process of 'computationally' determining whether a piece of writing is positive, negative or neutral. It's also known as opinion mining, deriving the opinion or attitude of a speaker. The proposed system is to analyze the sentiment of the people using python, twitter API, Text Blob (Library for processing text). As the results it helps to analysis the post with a better accuracy.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In the past few years, there has been a huge growth in the use of micro blogging platforms such as Twitter. Spurred by that growth, companies and media organizations are increasingly seeking ways to mine Twitter for information about what people think and feel about their products and services. Companies such as Twitratr (twitrratr.com), tweet feel (www.tweetfeel.com), and Social Mention (www.socialmention.com) are just a few who advertise Twitter sentiment analysis as one of their services.

While there has been a fair amount of research on how sentiments are expressed in genres such as online reviews and news articles, how sentiments are expressed given the informal language and message-length constraints of micro blogging has been much less studied. Features such as automatic part-of-speech tags and resources such as sentiment lexicons have proved useful for sentiment analysis in other domains, but will they also prove useful for sentiment analysis in Twitter? In this paper, we begin to investigate this question.

Another challenge of micro blogging is the incredible breadth of topic that is covered. It is not an exaggeration to say that people tweet about anything and everything. Therefore, to be able to build systems to mine Twitter sentiment about any given topic, we need a method for quickly identifying data that can be used for training. In this paper, we explore one method for building such data: using Twitter hashtags (e.g., #bestfeeling, #epicfail, #news) to identify positive, negative tweets to use for training three way sentiment classifiers.

The online medium has become a significant way for people to express their opinions and with social media, there is an abundance of opinion information available. Using sentiment analysis, the polarity of opinions can be found, such as positive, negative, or neutral by analyzing the text of the opinion. Sentiment analysis has been useful for companies to get their customer's opinions on their products predicting outcomes of elections, and getting opinions from movie reviews. The information gained from sentiment analysis is useful for companies making future decisions.

Many traditional approaches in sentiment analysis uses the bag of words method. The bag of words technique does not consider language morphology, and it could incorrectly classify two phrases of having the same meaning because it could have the same bag of words . The relationship between the collection of words is considered instead of the relationship between individual words. When determining the overall sentiment, the sentiment of each word is determined and combined using a function . Bag of words also ignores word order, which leads to phrases with negation in them to be incorrectly classified. Other techniques discussed in sentiment analysis include Naive Bayes, Maximum Entropy, and Support Vector Machines.

Sentiment analysis refers to the broad area of natural language processing which deals with the computational study of opinions, sentiments and emotions expressed in text. Sentiment Analysis (SA) or Opinion Mining (OM) aims at learning people's opinions, attitudes and emotions towards an entity. The entity can represent individuals, events or topics. An immense amount of research has been performed in the area of sentiment analysis. But most of them focused on classifying formal and larger pieces of text data like reviews. With the wide popularity of social networking and microblogging websites and an immense amount of data available from these resources, research projects on sentiment analysis have witnessed a gradual domain shift. The past few years have witnessed a huge growth in the use of microblogging platforms. Popular microblogging websites like Twitter have evolved to become a source of varied information. This diversity in the information owes to such microblogs being elevated as platforms where people post real time messages about their opinions on a wide variety of topics, discuss current affairs and share their experience on products and services they use in daily life. Stimulated by the growth of microblogging platforms, organizations are exploring ways to mine Twitter for information about how people are responding to their products and services. A fair amount of research has been carried out on how sentiments are expressed in formal text patterns such as product or movie reviews and news articles, but how sentiments are expressed given the informal language and message-length constraints of microblogging has been less explored.

CHAPTER 2

ABOUT PROJECT

2.1 EXISTING SYSTEM

The existing system of sentiment analysis of twitter focuses on mining tweets written in English. It uses sentiment analysis to classify specific tweets about two restaurants, KFC and McDonald's. It examined weather specific tweets is positive, negative. The system showed interest in seeing who people think is better in terms of how good /bad reviews are

2.1.1 DRAWBACKS

- The existing system did not classify tweets on a random searched person or a thing other than the 2 restaurants KFC and McDonalds.
- The existing system does not predict the percentage of positive tweets and negative tweets.
- The existing system did not display top positive and negative tweets.

2.2 PROPOSED SYSTEM

In this paper, we are going to analysis the micro blog called as Twitter, classify the "tweets" about a searched person/thing into positive, negative sentiment. In this system, we will also predict the positive tweets percentage and negative tweets percentage and to show top (positive, negative) tweets on a person/thing. we will also devise a program to show live streaming of tweets about a person or a hash tag using python.

2.3 SYSTEM REQUIREMENTS

1. Linux Operating System/Windows
2. Python Platform(IDE)
3. Modern Web Browser
4. Twitter API, Google API
5. LANGUAGE :PYTHON

CHAPTER 3

LITERATURE SURVEY

3.1 BARBOSA ET AL.(2010) designed a two phase automatic sentiment analysis method for classifying tweets. They classified tweets as objective or subjective and then in second phase, the subjective tweets were classified as positive or negative. The feature space used included retweets, hashtags, link, punctuation and exclamation marks in conjunction with features like prior polarity of words and POS.

3.2 BIFET AND FRANK(2010) used Twitter streaming data provided by Firehouse API,which gave all messages from every user which are publicly available in real-time. They experimented multinomial naive Bayes, stochastic gradient descent, and the Hoeffding tree. They arrived at a conclusion that SGDbased model, when used with an appropriate learning rate was the better than the rest used. Davidov et al.,(2010) proposed a approach to utilize Twitter user-defined hastags in tweets as a classification of sentiment type using punctuation, single words, n-grams and patterns as different feature types, which are then combined into a single feature vector for sentiment classification. They made use of K-Nearest Neighbor strategy to assign sentiment labels by constructing a feature vector for each example in the training and test set.

3.3 PO-WEI LIANG ET.AL.(2014) used Twitter API to collect twitter data. Their training data falls in three different categories (camera, movie, mobile). The data is labeled as positive, negative and non-opinions. Tweets containing opinions were filtered. Unigram Naive Bayes model was implemented and the Naive Bayes simplifying independence assumption was employed. They also eliminated useless features by using the Mutual Information and Chi square feature extraction method. Finally, the orientation of an tweet is predicted. i.e. positive or negative.

3.4 SAHAR A. EL RAHMAN :SENTIMENT ANALYSIS OF TWITTER DATA, IEEE 2019

In this paper, we used sentiment analysis to classify specific English tweets about two restaurants, KFC and McDonald's. our research was determining which one better than other, in specific we examined weather specific tweets is positive, negative, neutral. This paper focus on mining tweets written in English. We are interested in seeing who people think is better Mcdonalds or KFC in terms of how good/bad reviews are. Analyzing people's opinions and what they think about a product from their tweets on social media could be a valuable thing for any business. In our project, we extracted tweets from Twitter using R language. R is a programming language used for statistical computing and machine learning algorithms. In order to extract tweets from Twitter, Twitter API were used to create Twitter application and get authorization. In Rstudio which is an environment and graphical user interface for R, we installed necessary packages and libraries.

CHAPTER 4

SYSTEM DESIGN

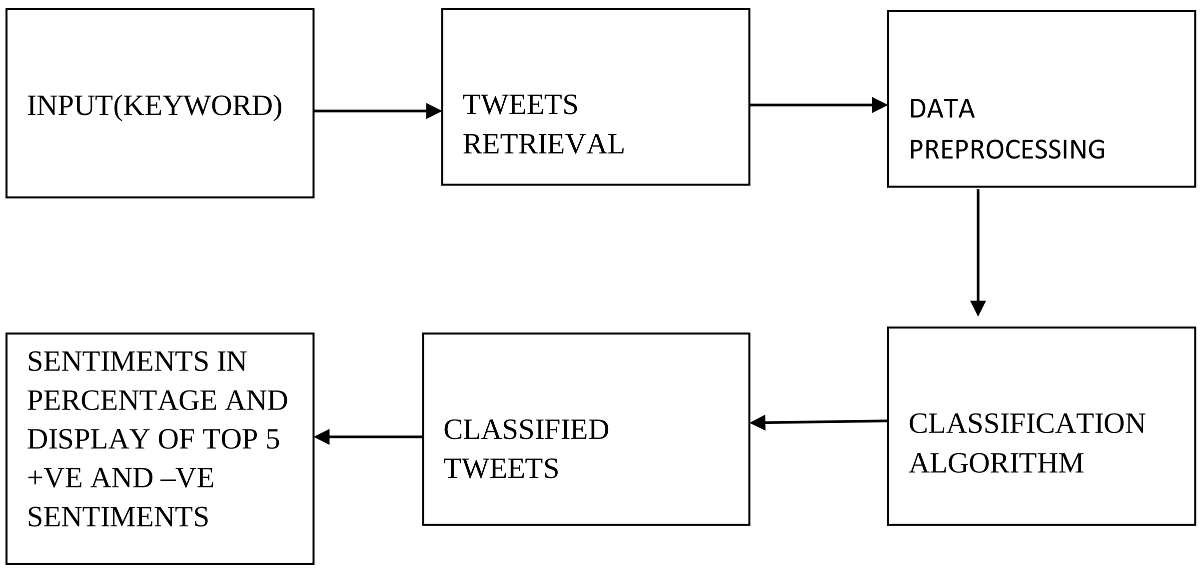

Fig 4.1 Block diagram of the model implementation 1

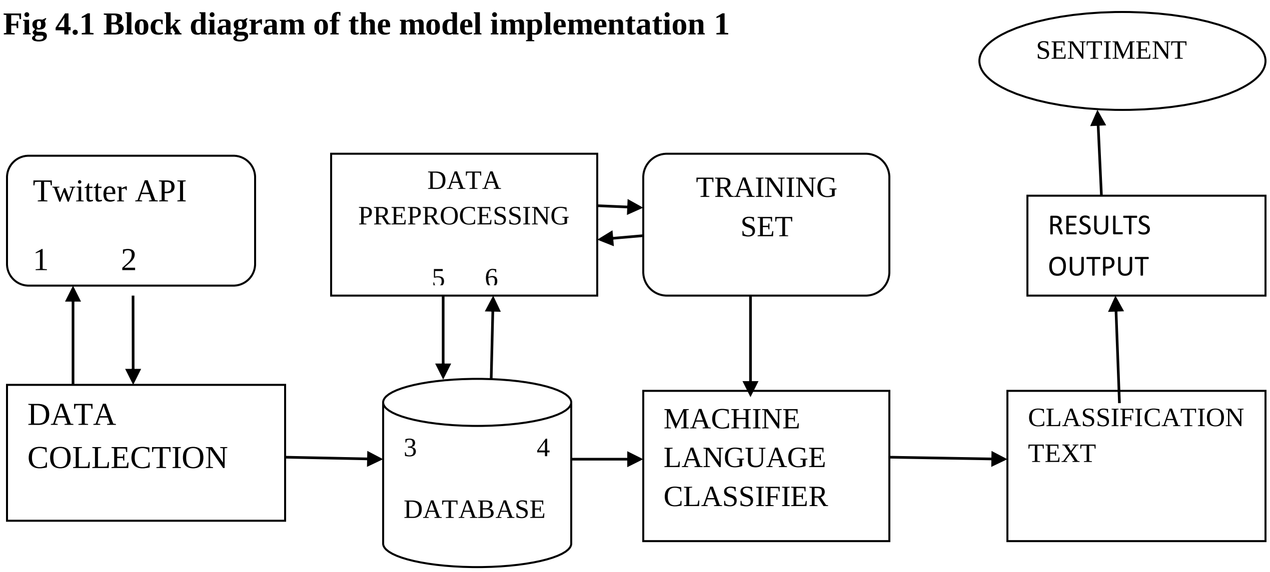

Fig 4.2 Block diagram of the model implementation 2(Decomposition model)

1) PROVIDES CREDENTIALS
2) TWEETS
3) WRITE TO DO
4) DATA TO BE CLASSIFY
5) READ TWEETS
6) WRITE TWEETS

CHAPTER 5

TOOLS AND TECHNOLOGY

5.1 PYTHON

In technical terms, Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development. It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options.

Python is relatively simple, so it's easy to learn since it requires a unique syntax that focuses on readability. Developers can read and translate Python code much easier than other languages. In turn, this reduces the cost of program maintenance and development because it allows teams to work collaboratively without significant language and experience barriers.

Additionally, Python supports the use of modules and packages, which means that programs can be designed in a modular style and code can be reused across a variety of projects. Once you've developed a module or package you need, it can be scaled for use in other projects, and it's easy to import or export these modules.

One of the most promising benefits of Python is that both the standard library and the interpreter are available free of charge, in both binary and source form. There is no exclusivity either, as Python and all the necessary tools are available on all major platforms. Therefore, it is an enticing option for developers who don't want to worry about paying high development costs.

That makes Python accessible to almost anyone. If you have the time to learn, you can create some amazing things with the language.

Python is a general-purpose programming language, which is another way to say that it can be used for nearly everything. Most importantly, it is an interpreted language, which means that the written code is not actually translated to a computer-readable format at runtime. Whereas, most programming languages do this conversion before the program is even run. This type of language is also referred to as a "scripting language" because it was initially meant to be used for trivial projects.

The concept of a "scripting language" has changed considerably since its inception, because Python is now used to write large, commercial style applications, instead of just banal ones. This reliance on Python has grown even more so as the internet gained popularity. A large majority of web applications and platforms rely on Python, including Google's search engine, YouTube, and the web-oriented transaction system of the New York Stock Exchange (NYSE). We know the language must be pretty serious when it's powering a stock exchange system.

Python can also be used to process text, display numbers or images, solve scientific equations, and save data. In short, it is used behind the scenes to process a lot of elements you might need or encounter on your device(s) - mobile included.

BENEFITS:

1) Python can be used to develop prototypes, and quickly because it is so easy to work with and read.

2) Most automation, data mining, and big data platforms rely on Python.

3) Python allows for a more productive coding environment than massive languages like C# and Java. Experienced coders tend to stay more organized and productive when working with Python

4) Python is easy to read, even if you're not a skilled programmer. Anyone can begin working with the language, all it takes is a bit of patience and a lot of practice. Plus, this makes it an ideal candidate for use among multi-programmer and large development teams.

5) Python powers Django, a complete and open source web application framework. Frameworks - like Ruby on Rails - can be used to simplify the development process.

6) It has a massive support base thanks to the fact that it is open source and community developed. Millions of like-minded developers work with the language on a daily basis and continue to improve core functionality. The latest version of Python continues to receive enhancements and updates as time progresses. This is a great way to network with other developers.

5.2 TWITTER API

The Twitter API allows you to access the features of Twitter without having to go through the website interface. This can be useful for doing things like posting tweets or sending directed messages in an automated way with scripts.

Say for example you are having a Twitter chat, and people have asked to receive a personal reminder tweet before it starts. If there were a hundred people, it would be a lot of work to manually send a tweet to everyone. However, with a list of usernames and a script that accesses Twitter through the API, you could automate sending the reminder tweets so it's accomplished quickly and easily.

Twitter is an information network and communication mechanism that produces more than 200 million tweets a day. The Twitter platform offers access to that corpus of data, via our APIs. Each API represents a facet of Twitter, and allows developers to build upon and extend their applications in new and creative ways. It's important to note that the Twitter APIs are constantly evolving, and developing on the Twitter Platform is not a one-off event.

5.3 SENTIMENT ANALYSIS TECHNIQUES

Sentiment Analysis (SA) elucidates users whether information or opinion regarding a certain product is positive, negative or neutral. Sentiment basically refers to any opinion or a feeling expressed by someone. Various organizations use this analysis to understand users' opinion for their products. For example, a particular e-commerce website can utilize sentiment analysis to discern if their products are being liked by the customers or not. The reviews for the products can be generalized into either positive or negative as well as neutral categories.

There are basically three techniques to perform Sentiment Analysis.

1. SA using machine learning.
2. SA using lexicon based techniques
3. SA using the above two techniques combined together.

1. Machine learning technique involves both supervised and unsupervised learning. 1.1 Unsupervised Learning is based on just inputs, without any mention of targets. It just relies on clustering. 1.2 Supervised Learning defines pre-specified targets which should be achieved, along with the inputs. Data set are trained to achieve significant outputs when encountered during decision-making.

2. Lexicon-Based Approaches: Lexicon based method assigns positive or negative polarity based on the sentiment of each word and then a dictionary is created. We can use a combining function, for example, sum or average to find out the general sentiment of a document.

3. Hybrid techniques combine both machine learning and lexicon based approaches to get better classification results. The dominant features of both these methods can be used to obtain steady results

There are 4 levels of sentiment analysis:

1) Document level: In this, whole document is classified as either positive or negative[6]. Respective words and sentences are checked for sentiments and are subsequently combined to find the sentiment polarity of the complete document.

2) Word level: Word level sentiment analysis utilizes adjectives and adverbs that define the sentiment of each word. Two methods that comment on sentiment at word level are : 1. Dictionary-based approaches 2. Corpus-based approaches

3) Sentence or phrase level: In this sentiment analysis, each sentence is categorized as either positive, or negative and may be neutral as well. If a sentence turns out to be neutral, it means there is no opinion. All the sentences can then be combined to find polarity of a paragraph or even complete document.

4) Feature-level or aspect-level: It helps to analyze what people are trying to suggest. Feature level tries to extract sentiment from the opinion directly. Various tasks involved in this are: i. Identify and extract object features on which opinion holder has commented. ii. Check the polarity of each opinion whether it is positive or negative. It can be even neutral. iii. Find feature synonym.

CHAPTER 6

SRS (SOFTWARE REQUIREMENT SPECIFICATION)

6.1 INTERNAL INTERFACE REQUIREMENT

Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem. Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.

Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.

Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here. The recent explosion in data pertaining to users on social media has created a great interest in performing sentiment analysis on this data using Big Data and Machine Learning principles to understand people's interests. This project intends to perform the same tasks. The difference between this project and other sentimnt analysis tools is that, it will perform real time analysis of tweets based on hashtags and not on a stored archive.

Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this ;software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.

The Product functions are: Collect tweets in a real time fashion i.e. , from the twitter live stream based on specified hashtags . Remove redundant information from these collected tweets. Store the formatted tweets in MongoDB database . Perform Sentiment Analysis on the tweets stored in the database to classify their nature viz. positive, negative and so on. Use a machine learning algorithm which will predict the 'mood' of the people with respect ot that topic. Summarize the major functions the product must perform or must let the user perform. Organize the functions to make them understandable to ;any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data ;flow diagram or object class diagram, is often effective.

Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy. Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

6.2 EXTERNAL INTERFACE REQUIREMENT

We classify External Interface in 4 types, those are:

User Interface: Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.

Hardware interface: Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.

Software Interface: Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.

Communication Interface: Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

6.3 NON FUNCTIONAL REQUIREMENT

Performance Requirements: If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

Safety Requirements: Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.

Security Requirements: Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.

Software Quality Attributes: Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

CHAPTER 7

MODULES

7.1 NAÏVE BAYES CLASSIFICATION

Many language processing tasks are tasks of classification, although luckily our classes are much easier to define than those of Borges. In this classification we present the naive Bayes algorithms classification, demonstrated on an important classification problem: text categorization, the task of classifying an entire text by assigning it a text categorization label drawn from some set of labels.

We focus on one common text categorization task, sentiment analysis, the ex-sentiment analysis traction of sentiment, the positive or negative orientation that a writer expresses toward some object. Are view of a movie, book, or product on the web expresses the author's sentiment toward the product, while an editorial or political text expresses sentiment toward a candidate or political action. Automatically extracting consumer sentiment is important for marketing of any sort of product, while measuring public sentiment is important for politics and also for market prediction.

Naive Bayes is a probabilistic classifier, meaning that for a document d, out of all classes $c \in C$ the classifier returns the class ˆ c which has the maximum posterior probability given the document. In Eq. 1 we use the hat notation to mean "our estimate of the correct class".

$c = \text{argmax } P(c|d) \quad \text{where } c \in C$

This idea of Bayesianinference has been known since the work of Bayes(1763),Bayesian inference and was first applied to text classification by Mosteller and Wallace (1964). The intuitionofBayesianclassificationistouseBayes'ruletotransformintootherprobabilities. Bayes' rule is presented in Eq; it gives us a way to break down any conditional probability $P(x|y)$ into three other probabilities:

$P(x|y) = P(y|x)P(x) / P(y)$

7.2 ALGORITM

The algoritham of sentiment analysis using Naive Bayes Classification:

function BOOTSTRAP(x,b) returns p-value(x)

Calculate δ(x)

for i = 1to b do

for j = 1to n do # Draw a bootstrap sample x∗(i) of size n

Select a member of x at random and add it to x∗(i)

Calculate δ(x∗(i))

For each x∗(i)

s←s + 1ifδ(x∗(i)) > 2δ(x)

p-value(x)≈ s b

return p-value(x)

• Many language processing tasks can be viewed as tasks of classification. learn to model the class given the observation.

• Textcategorization, in which an entire text is assigned a class from a finiteset, comprises such tasks as sentiment analysis, spam detection, email classification, and authorship attribution.

• Sentimentanalysisclassifiesatextasreflectingthepositiveornegativeorientation (sentiment) that a writer expresses toward some object.

• Naive Bayes is a generative model that make the bag of words assumption (positiondoesn’tmatter) and the conditional independence assumption(words are conditionally independent of each other given the class)

• Naive Bayes with binarized features seems to work better for many text classification tasks.

The TextBlob package for Python is a convenient way to do a lot of Natural Language Processing (NLP) tasks. For example:

From textblob import TextBlob

TextBlob(“not a very great calculation”).sentiment

This tells us that the English phrase “not a very great calculation” has a polarity of about -0.3, meaning it is slightly negative, and a subjectivity of about 0.6, meaning it is fairly subjective.

There are helpful comments like this one, which gives us more information about the numbers we're interested in:

Each word in the lexicon has scores for: # 1) polarity: negative vs. positive (-1.0 => +1.0) # 2) subjectivity: objective vs. subjective (+0.0 => +1.0) # 3) intensity: modifies next word? (x0.5 => x2.0)

The lexicon it refers to is in en-sentiment.xml, an XML document that includes the following four entries for the word “great”.

<word Form=”great” cornetto svnset id=”n_a-525317” wordnet id=”a-01123879” pos=”JJ” sense=”very good” polanty=”1.0” subjectivity=”1.0” intensity=”1.0” confidence=”0.9” />

<word Form=”great” wordnet id=”a-011238818” pos=”JJ” sense=”of major significance or importance” polanty=”1.0” subjectivity=”1.0” intensity=”1.0” confidence=”0.9” />

<word Form=”great” wordnet id=”a-01123883” pos=”JJ” sense=”relativity large in size or number or extent” polanty=”0.4” subjectivity=”0.2” intensity=”1.0” confidence=”0.9” />

<word Form=”great” wordnet id=”a-01677433” pos=”JJ” sense=”remarkable or out of the ordinary in degree or magnitude or effect” polanty=”0.8” subjectivity=”0.8” intensity=”1.0” confidence=”0.9” />

In addition to the polarity, subjectivity, and intensity mentioned in the comment above, there's also “confidence”, but I don't see this being used anywhere. In the case of “great” here it's all the same part of speech (JJ, adjective), and the senses are themselves natural language and not used.

When calculating sentiment for a single word, TextBlob uses a sophisticated technique known to Mathematicians as “averaging”.

```
TextBlob(“great").sentiment
```

```
## Sentiment(polarity=0.8, subjectivity=0.75)
```

At this point we might feel as if we're touring a sausage factory. That feeling isn't going to go away, but remember how delicious sausage is! Even if there isn't a lot of magic here, the results can be useful—and you certainly can't beat it for convenience.

TextBlob doesn't not handle negation, and that ain't nothing!

```
TextBlob("not great").sentiment
```

```
## Sentiment(polarity=-0.4, subjectivity=0.75)
```

Negation multiplies the polarity by -0.5, and doesn't affect subjectivity.

TextBlob also handles modifier words!

Recognizing “very” as a modifier word, TextBlob will ignore polarity and subjectivity and just use intensity to modify the following word:

```
TextBlob("very great").sentiment
```

```
## Sentiment(polarity=1.0, subjectivity=0.9750000000000001)
```

The polarity gets maxed out at 1.0, but you can see that subjectivity is also modified by “very” to become 0.75·1.3=0.9750.75·1.3=0.975.

Negation combines with modifiers in an interesting way: in addition to multiplying by -0.5 for the polarity, the inverse intensity of the modifier enters for both polarity and subjectivity.

```
TextBlob("not very great").sentiment
```

```
#Sentiment(polarity=-0.3076923076923077, subjectivity=0.5769230769230769)
```

polarity=−0.5·11.3·0.8≈−0.31polarity=−0.5·11.3·0.8≈−0.31

subjectivity=11.3·0.75≈0.58subjectivity=11.3·0.75≈0.58

TextBlob will ignore one-letter words in its sentiment phrases, which means things like this will work just the same way:

TextBlob("not a very great").sentiment

##Sentiment(polarity=-0.3076923076923077, subjectivity=0.5769230769230769)

And TextBlob will ignore words it doesn't know anything about:

TextBlob("not a very great calculation").sentiment

##Sentiment(polarity=-0.3076923076923077, subjectivity=0.5769230769230769)

TextBlob goes along finding words and phrases it can assign polarity and subjectivity to, and it averages them all together for longer text.

And while I'm being a little critical, and such a system of coded rules is in some ways the antithesis of machine learning, it is still a pretty neat system and I think I'd be hard-pressed to code up a better such solution.

CHAPTER 8

SOURCE CODE

8.1 PROGRAM FOR LIVE STREAMING TWEETS

```
import tweepy

from tweepy.streaming import StreamListener

from tweepy import OAuthHandler

from tweepy import Stream


consumer_key = '7j07BiWjfipDjjdBScb883fS8'

consumer_secret = 'EDhIgz4F2TOtXajTLP11sUjnx4bwdhPEugv72BfmS2DXCj8bkO'

access_token = '1008722416237834246-Y4yEdIUx2MziUi5Y9q53lpSTO88zdn'

access_token_secret = 'OlwwLwj7omhAvlAf0fHtIux44IRRgEH3AhmwixvQJB08n'


class TwitterStreamer():


    def __init__(self):

        pass

    def stream_tweets(self, fetched_tweets_filename, hash_tag_list):

          listener = StdOutListener(fetched_tweets_filename)

        auth = OAuthHandler(consumer_key, consumer_secret)

        auth.set_access_token(access_token, access_token_secret)
```

```
        stream = Stream(auth, listener)

        stream.filter(track=hash_tag_list)


class StdOutListener(StreamListener):


    def __init__(self, fetched_tweets_filename):

        self.fetched_tweets_filename = fetched_tweets_filename


    def on_data(self, data):

        try:

            print(data)

            with open(self.fetched_tweets_filename, 'a') as tf:

                tf.write(data)

            return True

        except BaseException as e:

            print("Error on_data %s" % str(e))

        return True



    def on_error(self, status):

        print(status)
```

```
if __name__ == '__main__':

    hash_tag_list = ["donal trump", "hillary clinton", "barack obama", "bernie sanders"]

    fetched_tweets_filename = "tweets.txt"


    twitter_streamer = TwitterStreamer()

    twitter_streamer.stream_tweets(fetched_tweets_filename, hash_tag_list).
```

8.2 PROGRAM FOR RECEIVING POSITIVE AND NEGATIVE TWEETS AND ITS PERCENATAGE

```
import re

import tweepy

from tweepy import OAuthHandler

from textblob import TextBlob


class TwitterClient(object):

    '''

    Generic Twitter Class for sentiment analysis.

    '''

    def __init__(self):

        '''

        Class constructor or initialization method.
```

```
    '''
    # keys and tokens from the Twitter Dev Console
    consumer_key = '7j07BiWjfipDjjdBScb883fS8'
    consumer_secret = 'EDhIgz4F2TOtXajTLP11sUjnx4bwdhPEugv72BfmS2DXCj8bkO'
    access_token = '1008722416237834246-Y4yEdIUx2MziUi5Y9q53lpSTO88zdn'
    access_token_secret = 'OlwwLwj7omhAvlAf0fHtIux44IRRgEH3AhmwixvQJB08n'

    # attempt authentication
    try:
        # create OAuthHandler object
        self.auth = OAuthHandler(consumer_key, consumer_secret)
        # set access token and secret
        self.auth.set_access_token(access_token, access_token_secret)
        # create tweepy API object to fetch tweets
        self.api = tweepy.API(self.auth)
    except:
        print("Error: Authentication Failed")

def clean_tweet(self, tweet):
    '''
    Utility function to clean tweet text by removing links, special characters
```

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

    def get_tweet_sentiment(self, tweet):
        '''
        Utility function to classify sentiment of passed tweet
        using textblob's sentiment method
        '''
        # create TextBlob object of passed tweet text
        analysis = TextBlob(self.clean_tweet(tweet))
        # set sentiment
        if analysis.sentiment.polarity > 0:
            return 'positive'
        elif analysis.sentiment.polarity == 0:
            return 'neutral'
        else:
            return 'negative'

    def get_tweets(self, query, count = 10):
        '''
```

```
Main function to fetch tweets and parse them.
'''
# empty list to store parsed tweets
tweets = []

try:
    # call twitter api to fetch tweets
    fetched_tweets = self.api.search(q = query, count = count)

    # parsing tweets one by one
    for tweet in fetched_tweets:
        # empty dictionary to store required params of a tweet
        parsed_tweet = {}

        # saving text of tweet
        parsed_tweet['text'] = tweet.text
        # saving sentiment of tweet
        parsed_tweet['sentiment'] = self.get_tweet_sentiment(tweet.text)

        # appending parsed tweet to tweets list
        if tweet.retweet_count > 0:
```

```
                # if tweet has retweets, ensure that it is appended only once
                if parsed_tweet not in tweets:
                    tweets.append(parsed_tweet)
            else:
                tweets.append(parsed_tweet)

        # return parsed tweets
        return tweets

    except tweepy.TweepError as e:
        # print error (if any)
        print("Error : " + str(e))

def main():
    # creating object of TwitterClient Class
    api = TwitterClient()
    # calling function to get tweets
    tweets = api.get_tweets(query = 'DONALD TRUMP', count = 200)

    # picking positive tweets from tweets
    ptweets = [tweet for tweet in tweets if tweet['sentiment'] == 'positive']
```

```
    # percentage of positive tweets

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

    # picking negative tweets from tweets

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

    # percentage of negative tweets

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



    print("\n\nPositive tweets:")

    for tweet in ptweets[:5]:

        print(tweet['text'])

    print("\n\nNegative tweets:")

    for tweet in ntweets[:5]:

        print(tweet['text'])




if __name__ == "__main__":

    # calling main function

    main()
```

8.3 PROGRAM TO CLASSIFY TWEETS ON A RANDOM SEARCHED PERSON OR A THING

```
import tweepy

from textblob import TextBlob


consumer_key = 'FSQkZVtwbLGi7f2UNWFpR4tNo'

consumer_secret = 'Afsz81V8sTrBzzbMaNufmxFdbqPoo0DssXhcSNZRPrBnl84m6G'

access_token = '1008722416237834246-ZVt8wZWdXEZvDTZ8Ygm5ZsXliGKX3p'

access_token_secret = 'Xjsvpr3TBalNQHz6opr977jxYycTl6X8yEJa5hVkZVEv8'


auth = tweepy.OAuthHandler(consumer_key, consumer_secret)


auth.set_access_token(access_token, access_token_secret)


api = tweepy.API(auth)


public_tweets = api.search('Narendra modi')


for tweet in public_tweets:

        print(tweet.text.encode('UTF-8').decode('UTF-8'))

        analysis = TextBlob(tweet.text)
```

```
    print(analysis.sentiment)

    if analysis.sentiment[0]>0:

        print ('Positive')

    else:

        print ('Negative')

    print("")



input("Press enter to close")
```

CHAPTER 9

SCREENSHOTS

FIG 9.1 **Live streaming of tweets**

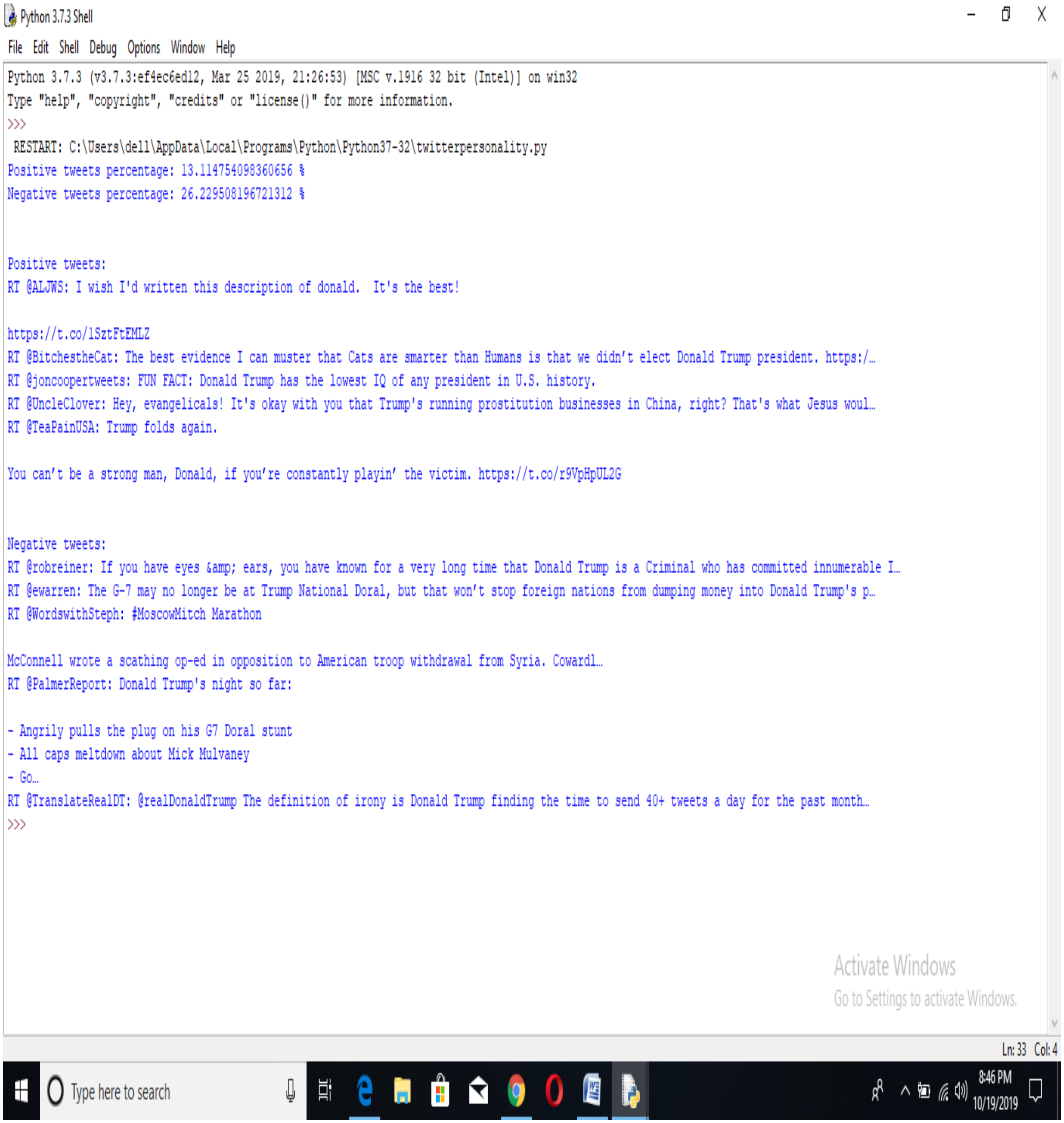

FIG 9.2 Receiving Positive and Negative Tweets and its percentage

(e.g.: DONALD TRUMP)

Python 3.7.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\dell\AppData\Local\Programs\Python\Python37-32\twitterpersonality.py
Positive tweets percentage: 23.404255319148938 %
Negative tweets percentage: 6.382978723404255 %



Positive tweets:
RT @navinbhansali: PM Modi was gracious in congratulating me, economic slowdown not India specific: Esther Duflo https://t.co/Z6ITbYDY76
PM Narendra Modi exchanged the thoughts and views with film personalities and cultural icons on spreading the thoug… https://t.co/SGh5JAzYnw
RT @kapsology: ₹2,00,000 Crores of bank frauds under Narendra Modi since he came to power.

Can you imagine how much money is this? This is…
@upasanakonidela @narendramodi Dearest Narendra Modi ji. We in the south of India admire you and r are proud to hav… https://t.co/Z2e6FbvXTH
RT @dhume: .@tavleen_singh: There are only two kinds of really rich Indians—wealth creators and looters. Modi's draconian tax policies ha…
RT @ZeeNews: How a Chinese movie became the talking point when PM Narendra Modi met top film stars

https://t.co/ST26GpO5dW
RT @OpIndia_com: PM Modi was gracious in congratulating me, economic slowdown not India specific: Esther Duflo on India, PM Modi and the Ec…
Liked on YouTube: Narendra Modi attack on the Congress party. https://t.co/ry6KEe1FBV https://t.co/AbmVUBIw7t
How a Chinese movie became the talking point when PM Narendra Modi met top film stars https://t.co/hEmcV23xPV
RT @airnewsalerts: PM Narendra Modi, released 4 cultural videos today in Delhi marking the #150YrsofMahatma Gandhi.

The event was attended…



Negative tweets:
@_pallavighosh @PMOIndia @iamsrk sorry, but this is utterly butterly ridiculous... Here is what Shri Narendra Modi,… https://t.co/UBihyR9e79
RT @EruditePeacenik: 55 years later Narendra Modi is narrating a cock and bull story on Kashmir in HARYANA. Will this sly propaganda delive…
RT @ANI: Delhi: Shahrukh Khan, Aamir Khan, Kangana Ranaut and other members of film fraternity with Prime Minister Narendra Modi after an i…
>>>
```


Ln: 32 Col: 4

Type here to search

8:48 PM
10/19/2019

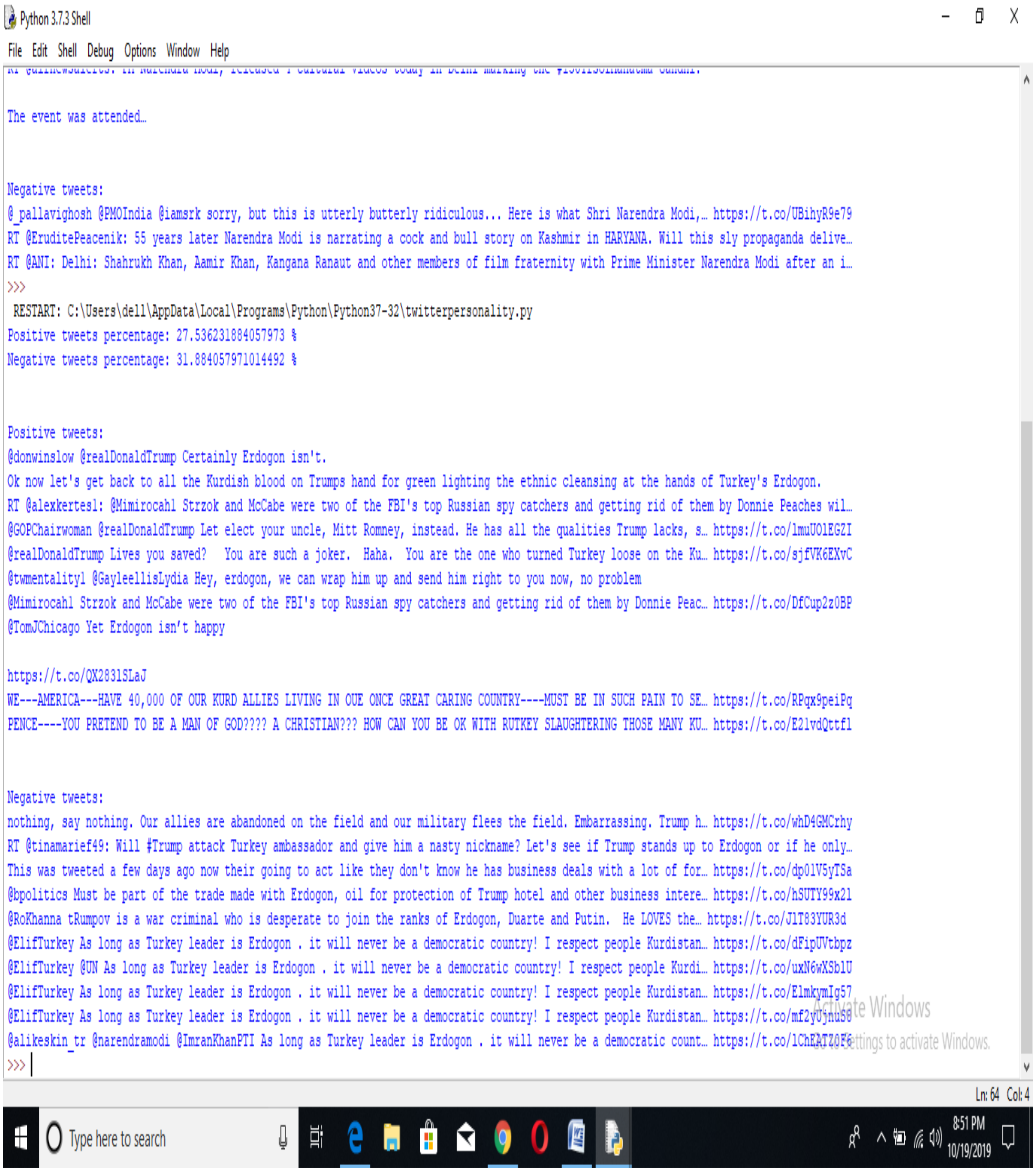

FIG 9.3 Receiving Positive and Negative Tweets and its percentage

(e.g.: ERDOGON)

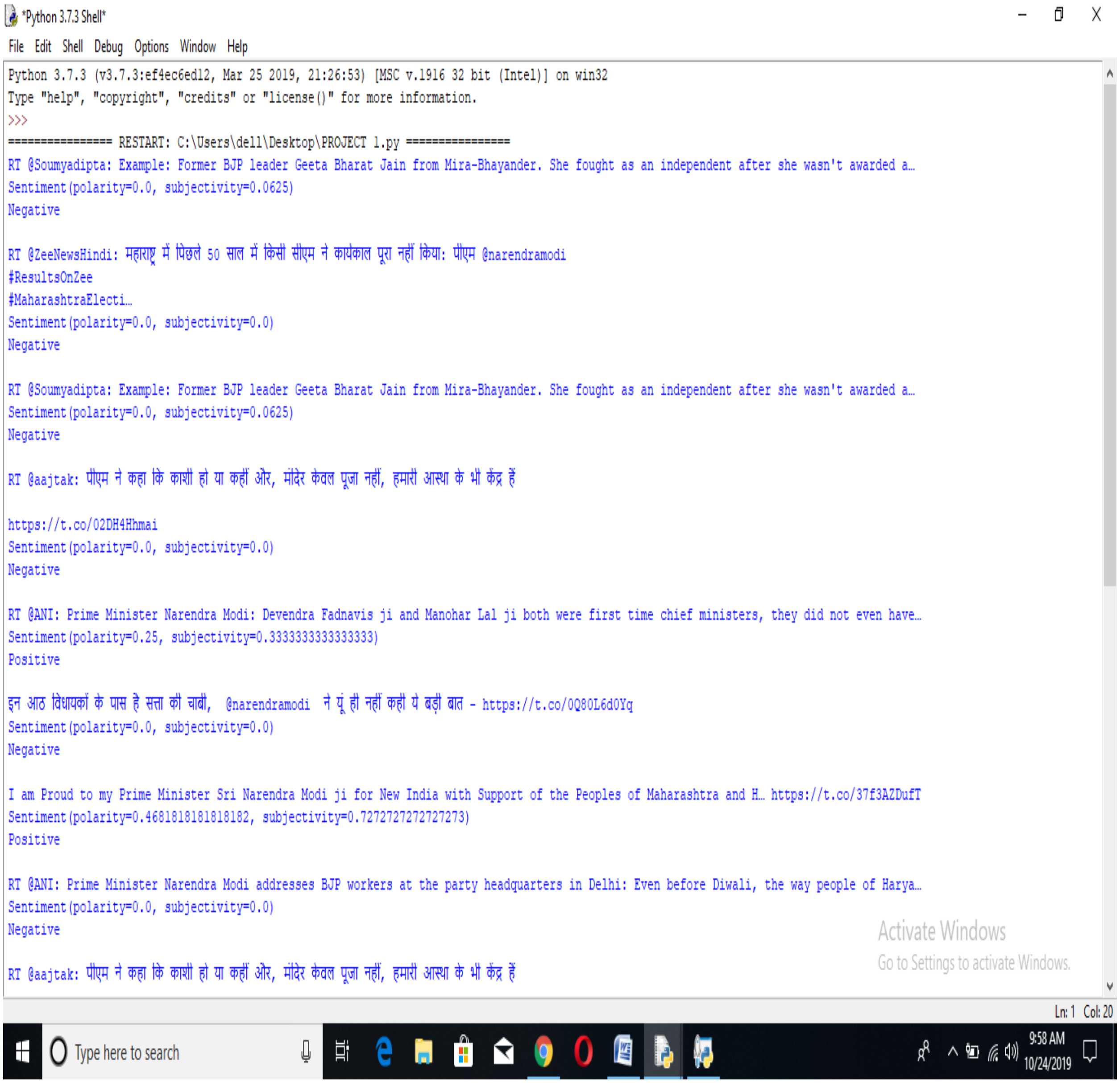

FIG 9.4 To classify tweets on a random searched person or a thing

Based on its sentiment

(e.g.: NARENDRA MODI)

CHAPTER 10

CONCLUSION AND FUTURE SCOPE

There are various types of Machine Learning Algorithms like Naive Bayes, Max entropy that may not produce accurate results for either of unigrams, bigrams or weighted unigrams. Support Vector Machines (SVM) is termed as best data classification technique. Sentiment Analysis on twitter data is nothing different from those techniques on other genres. For further in the future these techniques can be explored to rich linguistic analysis like topic modeling and semantic analysis.

Semantics: The comprehensive sentiment of a tweet is classified by the algorithms. Semantic role labeler can be used which indicates which noun is associated with the verb and accordingly the classification occurs.

Internationalization: Here the focus is only on English tweets but Twitter has a large amount of international audience. This approach should be used to classify sentiment with a language specific positive/negative keyword list in other languages.

REFERENCES

[1] Efthymios Kouloumpis and Johanna Moore,IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4, No 3, July 2012

[2] S. Batra and D. Rao, ”Entity Based Sentiment Analysis on Twitter”, Stanford University,2010

[3] Saif M.Mohammad and Xiaodan zhu ,Sentiment Analysis on of social media texts:,2014

[4]Ekaterina kochmar,University of Cambridge,at the Cambridge coding Academy Data Science.2016

[5] Manju Venugopalan and Deepa Gupta ,Exploring Sentiment Analysis on Twitter Data, IEEE 2015

[6] Brett Duncan and Yanqing Zhang, Neural Networks for Sentiment Analysis on Twitter.2017

[7] Afroze Ibrahim Baqapuri, Twitter Sentiment Analysis: The Good the Bad and the OMG!, Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media.2011

[8] Sahar A. El Rahman Feddah Alhumaidi AlOtaibi Wejdan Abdullah AlShehri :Sentiment Analysis of Twitter Data, IEEE 2019