NATURAL LANGUAGE PROCESSING THROUGH SENTIMENT ANALYSIS

A Major project report submitted in fulfilment of the requirements for the award of the degree of

B.TECH(CSE) + M.TECH(ICT)

with specialization in

SOFTWARE ENGINEERING

Submitted by:

Radhika Gupta
Roll no.:15/ICS/067

Varnika Gupta
Roll no.:15/ICS/058

Kaavya Singh
Roll no.:15/ICS/075

Supervised by:
Dr. Pradeep Tomar
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY,
GAUTAM BUDDHA UNIVERSITY, GREATER NOIDA-201312
GAUTAM BUDDHA NAGAR, UTTAR PRADESH, INDIA.

MAY, 2019

ACKNOWLEDGEMENTS

It gives us immense pleasure in presenting this project report on "**Natural language processing through Sentiment Analysis**". The writing of this project has been one of the significant academic challenges we have faced and without the support, patience and guidance of the people involved in this project, this would not have been completed. It is to them that we owe our deepest gratitude.

The success of this project is the outcome of sheer hard work and determination put in together by our team with the help of our project mentor and guide.

We hereby take this opportunity to add a special note of thanks to **Dr. Pradeep Tomar.** Without his insight and constant support it would not have been possible to kick start this project and to reach the completion of this report.

Kaavya Singh 15/ICS/075

Radhika Gupta 15/ICS/067

Varnika Gupta 15/ICS/058

ABSTRACT

Sentiment analysis refers to the task of natural language processing to determine whether a piece of text contains some subjective information and what subjective information it expresses, i.e., whether the attitude behind this text is positive, negative or neutral. Understanding the opinions behind user-generated content automatically is of great help for commercial and political use, among others. The task can be conducted on different levels, classifying the polarity of words, sentences or entire documents.

In today's world, Social Networking websites like Twitter, Facebook, Tumblr etc. play a very significant role. Twitter is a micro-blogging platform which provides a tremendous amount of data which can be used for various applications of Sentiment Analysis like predictions, reviews, elections, marketing etc. Here, Sentiment Analysis can also be defined as the process of extracting information from a large amount of data, and classifying them into sentiments.

This project performs Sentiment Analysis using Python. Python is a simple yet powerful, high level, interpreted and dynamic programming language, which is well known for its functionality of processing natural language data using NLTK. NLTK is a library of Python, which provides a base for building programs and classification of data. It also provides a graphical demonstration for representing various results or trends and it also provides sample data to train and test various classifiers respectively.

Three major steps are followed in this project:

- Authorize twitter API client.
- Make a GET request to Twitter API to fetch tweets for a particular query.
- Parse the tweets. Classify each tweet as positive, negative or neutral.

The aim of this project is to develop a functional classifier for accurate and automatic sentiment classification of an unknown tweet stream and to categorize the data as positive, weakly positive, strongly positive, negative, weakly negative, strongly negative and neutral. The user is asked to provide the keyword that is to be looked for, then asked to mention the number the tweets among which the sentiment of that particular word is to be generated. Next, the result is displayed in the form of a pie chart using the Mathplotlib library.

LIST OF ABBREVIATIONS

NLTK: Natural Language Processing Toolkit

NLP: Natural Language Processing

CSV: Comma Separated Values

RE: Regular Expressions

API: Application Programming Interface

RAM: Random Access Memory

OS: Operating System

SRS: System Requirements Specifications

UML: Unified Modeling Language

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

Introduction

1.1 Introduction

The opinions of others have a significant influence in daily decision-making process. These decisions range from buying a product such as a smart phone to making investments to choosing a school—all decisions that affect various aspects of our daily life. Before the Internet, people would seek opinions on products and services from sources such as friends, relatives, or consumer reports

Similarly, organizations use surveys, opinion polls, and social media as a mechanism to obtain feedback on their products and services. Sentiment analysis or opinion mining is the computational study of opinions, sentiments, and emotions expressed in text. The use of sentiment analysis is becoming more widely leveraged because the information it yields can result in the monetization of products and services. For example, by obtaining consumer feedback on a marketing campaign, an organization can measure the campaign's success or learn how to adjust it for greater success. Product feedback is also helpful in building better products, which can have a direct impact on revenue, as well as comparing competitor offerings.

1.2 Problem Identification

- A major benefit of social media is that the good and bad things people say about the particular brand or personality can be seen.
- The bigger the company gets, the difficult it becomes to keep a handle on how everyone feels about the brand. For large companies with thousands of daily mentions on social media, news sites and blogs, it's extremely difficult to do this manually.
- To combat this problem, sentiment analysis software are necessary. These softwares can be used to evaluate the people's sentiment about particular brand or personality.

1.3 Theoretical Background

This section defines the concepts involved in the making of the project.

1.3.1 Python

Python is a high level, interpreted programming language, created by Guido van Rossum. The language is very popular for its code readability and compact line of codes. It uses white space inundation to delimit blocks.

Python provides a large standard library which can be used for various applications for example natural language processing, machine learning, data analysis etc. It is favored for complex projects, because of its simplicity, diverse range of features and its dynamic nature.[1]

1.3.2 Natural Language Processing Toolkit

NLTK is a library in python, which provides the base for text processing and classification. Operations such as tokenization, tagging, filtering, text manipulation can be performed with the use of NLTK. The NLTK library also embodies various trainable classifiers (example – Naïve Bayes Classifier).

NLTK library is used for creating a bag-of words model, which is a type of unigram model for text. In this model, the number of occurrences of each word is counted. The data acquired can be used for training classifier models. The sentiment of the entire tweets is computed by assigning subjectivity score to each word using a sentiment lexicon.[1]

1.3.3 Tweepy

Tweepy is open-sourced, hosted on GitHub and enables Python to communicate with Twitter platform and uses its API.

Tweepy supports accessing Twitter via Basic Authentication and the newer method, OAuth. Twitter has stopped accepting Basic Authentication so OAuth is now the only way to use the Twitter API. Here is the sample of how to access the Twitter API using Tweepy with OAuth.

```
import tweepy
# Consumer keys and access tokens, used for OAuth
consumer_key = '7EyzTcAkINVS3T2pb165'
consumer_secret = 'a44R7WvbMW7L8I656Y4l'
access_token = 'z00Xy9AkHwp8vSTJ04L0'
access_token_secret = 'A1cK98w2NXXaCWMqMW6p'
# OAuth process, using the keys and tokens
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
# Creation of the actual interface, using authentication
api = tweepy.API(auth)
# Sample method, used to update a status
api.update_status('Hello Python Central!')
```

Tweepy provides access to the well documented Twitter API. With tweepy, it is possible to get any object and use any method that the official Twitter API offers. For example, a user object has its documentation at https://dev.twitter.com/docs/platform-objects/users and following those guidelines, tweepy can get the appropriate information.[7]

```
# Creates the user object. The me() method returns the user whose authentication keys were
```

```
used.
user = api.me()
print('Name: ' + user.name)
print('Location: ' + user.location)
print('Friends: ' + str(user.friends_count))
```

Gives us the following output:

```
Name: Ahmet Novalic
Location: Gradacac,Bih
Friends: 59
```

1.3.4 TextBlob

The TextBlob package for Python is a convenient way to do a lot of NLP tasks. For example:

```
from textblob import TextBlob
TextBlob("not a very great calculation").sentiment
## Sentiment(polarity=-0.3076923076923077, subjectivity=0.5769230769230769)
```

This tells 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.

After digging a bit, it is found that the main default sentiment calculation is defined in _text.py, which gives credit to the pattern library.

There are helpful comments like this one, which gives more information about the numbers:

```
# 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_synset_id="n_a-525317" wordnet_id="a-01123879" pos="JJ" sense="very good" polarity="1.0" subjectivity="1.0" intensity="1.0" confidence="0.9" />
<word form="great" wordnet_id="a-01278818" pos="JJ" sense="of major significance or importance" polarity="1.0" subjectivity="1.0" intensity="1.0" confidence="0.9" />
<word form="great" wordnet_id="a-01386883" pos="JJ" sense="relatively large in size or number or extent" polarity="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" polarity="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". 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. To simplify for readability:

```
word   polarity  subjectivity  intensity

great       1.0           1.0        1.0
great       1.0           1.0        1.0
great       0.4           0.2        1.0
great       0.8           0.8        1.0
```

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)
```

TextBlob doesn't not handle negation.

```
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. Here's the summarized record for "very" from the lexicon:

```
word   polarity  subjectivity  intensity
very        0.2           0.3        1.3
```

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 we 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.[8]

1.3.5 Python Libraries

The following python libraries are used in sentiment analysis.

1.3.5.1 Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.[9]

Installation:

Windows, Linux and macOS distributions have matplotlib and most of its dependencies as wheel packages. Run the following command to install matplotlib package.

```
python -mpip install -U matplotlib
```

Importing matplotlib :

```
from matplotlib import pyplot as plt
or
import matplotlib.pyplot as plt
```

Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information. Some of the sample plots are covered here.

Line plot :

```
# importing matplotlib module
from matplotlib import pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot
plt.plot(x,y)
# function to show the plot
plt.show()
```

Output :

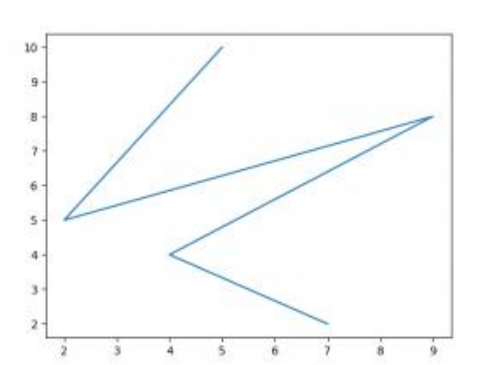

Figure 1.1: Line Plot

Bar plot :

```
# importing matplotlib module
from matplotlib import pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]
```

```
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot the bar
plt.bar(x,y)
# function to show the plot
plt.show()
```

Output:

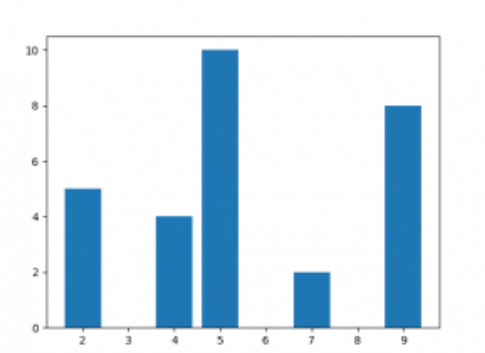

Figure 1.2: Bar Plot

Histogram :

```
# importing matplotlib module
from matplotlib import pyplot as plt
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot histogram
plt.hist(y)
# Function to show the plot
plt.show()
```

Output :

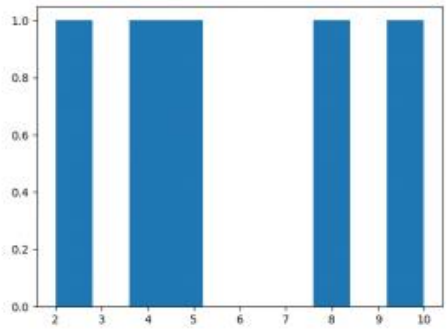

Figure 1.3: Histogram

Scatter Plot :

```
# importing matplotlib module
from matplotlib import pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot scatter
plt.scatter(x, y)
# function to show the plot
plt.show()
```

Output :

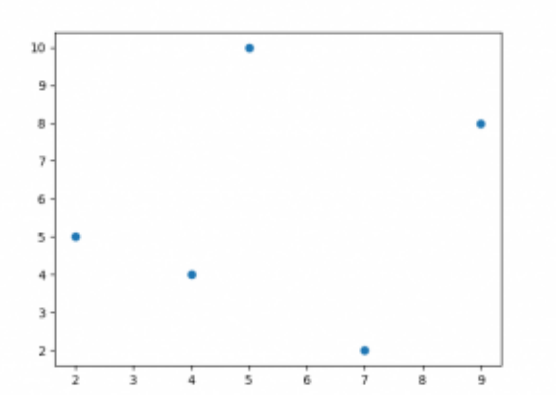

Figure 1.4: Scatter Plot

Pie chart:

```
import matplotlib.pyplot as plt
# Data to plot
labels = 'Python', 'C++', 'Ruby', 'Java'
sizes = [215, 130, 245, 210]
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']
explode = (0.1, 0, 0, 0)  # explode 1st slice
# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140
```

```
plt.axis('equal')
plt.show()
```

Output:

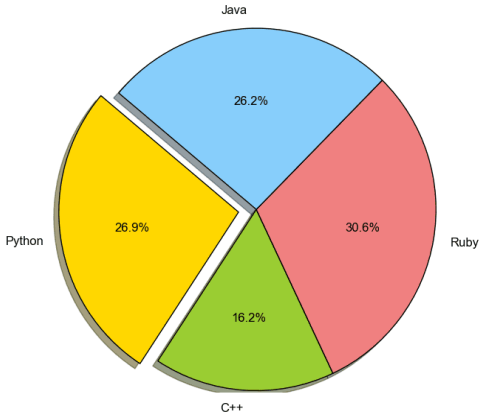

Figure 1.5: Pie Chart 1

```
import matplotlib.pyplot as plt
labels = ['Cookies', 'Jellybean', 'Milkshake', 'Cheesecake']
sizes = [38.4, 40.6, 20.7, 10.3]
colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']
patches, texts = plt.pie(sizes, colors=colors, shadow=True, startangle=90)
plt.legend(patches, labels, loc="best")
plt.axis('equal')
plt.tight_layout()
plt.show()
```

Output:

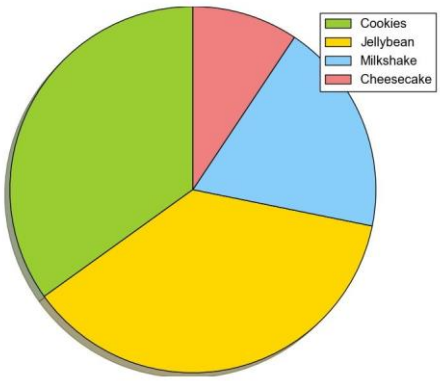

Figure 1.6: Pie Chart 2

1.3.5.2 Comma Separated Values

A CSV file is a type of plain text file that uses specific structuring to arrange tabular data. Because it's a plain text file, it can contain only actual text data—in other words, printable ASCII or Unicode characters.[10]

The structure of a CSV file is given away by its name. Normally, CSV files use a comma to separate each specific data value. Here's what that structure looks like:

```
column 1 name,column 2 name, column 3 name
first row data 1,first row data 2,first row data 3
second row data 1,second row data 2,second row data 3
...
```

1.3.5.3 Regular Expression

A RE is a sequence of characters that defines a search pattern. For example,

```
^a...s$
```

The above code defines a RE pattern. The pattern is**:** any five letter string starting with a and ending with s.

Python has a module named re to work with RegEx. Here's an example:

```
import re
pattern = '^a...s$'
test_string = 'abyss'
result = re.match(pattern, test_string)
if result:
  print("Search successful.")
else:
  print("Search unsuccessful.")
```

Here, we used re.match() function to search pattern within the test_string. The method returns a match object if the search is successful. If not, it returns None.

There are other several functions defined in the re module to work with RegEx.[11]

1.3.5.4 Sys Module

The sys module provides information about constants, functions and methods of the Python interpreter. dir(system) gives a summary of the available constants, functions and methods. Another possibility is the help() function. Using help(sys) provides valuable detail information.[12]
The module sys informs e.g. about the maximal recursion depth (sys.getrecursionlimit()) and provides the possibility to change (sys.setrecursionlimit())
The current version number of Python can be accessed as well:

```
>>> import sys
>>> sys.version
'2.6.5 (r265:79063, Apr 16 2010, 13:57:41) \n[GCC 4.4.3]'
>>> sys.version_info
(2, 6, 5, 'final', 0)
>>>
```

1.3.6 Data Pre-processing

• Pre-processing refers to the transformations applied to the data before feeding it to the algorithm.

• Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.[2]

1.3.6.1 Preprocessing Techniques

Data can be pre-processed using several techniques as discussed here −

- Mean removal
- Scaling
- Normalization

- Binarization
- One Hot Encoding
- Label Encoding

1.3.7 Sentiment Analysis

Sentiment Analysis is defined as the task of finding the opinions of authors about specific entities. The decision making process of people is affected by the opinions formed by thought leaders and ordinary people. When a person wants to buy a product online he/she will typically start by searching for reviews and opinions written by other people on the various offerings.

Sentiment analysis is contextual mining of text which identifies and extracts subjective information in source material, and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations. However, analysis of social media streams is usually restricted to just basic sentiment analysis and count based metrics. This is akin to just scratching the surface and missing out on those high value insights that are waiting to be discovered.

There is a huge explosion today of 'sentiments' available from social media including Twitter, Facebook, messages boards, blogs and user forums. These snippets of text are a gold mine for companies and individuals that want to monitor their reputation and get timely feedback about their products and actions.

Sentiment analysis offers these organisations the ability to monitor the different social media sites in real time and act accordingly. Marketing managers, PR firms, campaign managers, politicians and even equity investors and online shoppers are the direct beneficiaries of sentiment analysis technology.

Sentiment analysis can occur at different levels: document level, sentence level or aspect/feature level.

1.3.7.1 Document Level Classification

In this process, sentiment is extracted from the entire review, and a whole opinion is classified based on the overall sentiment of the opinion holder. The goal is to classify a review as positive, negative, or neutral.

Example:"I bought an iPhone a few days ago. It is such a nice phone, although a little large. The touch screen is cool. The voice quality is clear too. I simply love it!"

Is the review classification positive or negative? Document level classification works best when the document is written by a single person and expresses an opinion/sentiment on a single entity.

1.3.7.2 Sentence Level Classification

This process usually involves two steps:

- Subjectivity classification of a sentence into one of two classes: objective and subjective
- Sentiment classification of subjective sentences into two classes: positive and negative

An objective sentence presents some factual information, while a subjective sentence expresses personal feelings, views, emotions, or beliefs. Subjective sentence identification can be achieved through different methods such as Naïve Bayesian classification. However, just knowing that sentences have a positive or negative opinion is not sufficient. This is an intermediate step that helps filter out sentences with no opinions and helps determine to an extent if sentiments about entities and their aspects are positive or negative. A subjective sentence may contain multiple opinions and subjective and factual clauses.

Example:“iPhone sales are doing well in this bad economy.”

Sentiment classification at both the document and sentence levels are useful, but they do not find what people like or dislike, nor do they identify opinion targets.

1.3.7.3 Aspect/Feature Level Classification

In this process, the goal is to identify and extract object features that have been commented on by the opinion holder and determine whether the opinion is positive, negative, or neutral. Feature synonyms are grouped, and a feature-based summary of multiple reviews is produced.[3]

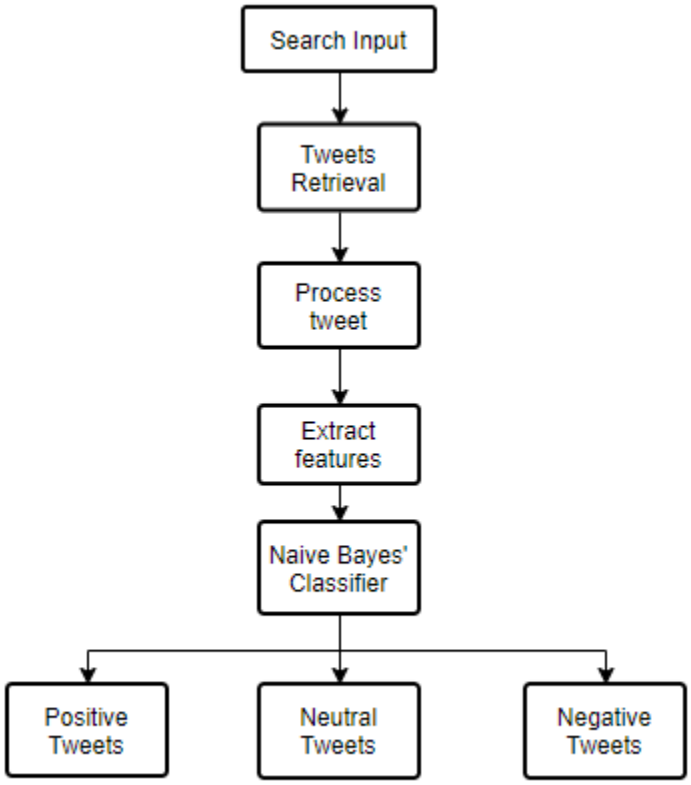

Figure 1.7: Workflow of the project

1.3.8 Data Set

A data set (or dataset) is a collection of data. Most commonly a data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the data set in question. The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set. Each value is known as a datum. The data set may comprise data for one or more members, corresponding to the number of rows.

1.3.8.1 Training Data

The observations in the training set form the experience that the algorithm uses to learn. In supervised learning problems, each observation consists of an observed output variable and one or more observed input variables.

1.3.8.2 Test Data

The test set is a set of observations used to evaluate the performance of the model using some performance metric. It is important that no observations from the training set are included in the test set. If the test set does contain examples from the training set, it will be difficult to

assess whether the algorithm has learned to generalize from the training set or has simply memorized it.

A program that generalizes well will be able to effectively perform a task with new data. In contrast, a program that memorizes the training data by learning an overly complex model could predict the values of the response variable for the training set accurately, but will fail to predict the value of the response variable for new examples. Memorizing the training set is called over-fitting. A program that memorizes its observations may not perform its task well, as it could memorize relations and structures that are noise or coincidence. Balancing memorization and generalization, or over-fitting and under-fitting, is a problem common to many machine learning algorithms. Regularization may be applied to many models to reduce over-fitting.[4]

1.3.9 Supervised Learning

Supervised learning is commonly used in real world applications, such as face and speech recognition, products or movie recommendations, and sales forecasting. Supervised learning can be further classified into two types - Regression and Classification.

Regression trains on and predicts a continuous-valued response, for example predicting real estate prices.

Classification attempts to find the appropriate class label, such as analyzing positive/negative sentiment, male and female persons, benign and malignant tumors, secure and unsecure loans etc.

Classification is a machine learning technique that uses known data to determine how the new data should be classified into a set of existing categories.[5]

1.3.10 Naive Bayes' Classifier Technique

Classification techniques include Naive Bayes Classifier, which is a simple technique for constructing classifiers. It is not one algorithm for training such classifiers, but a group of algorithms. A Bayes classifier constructs models to classify problem instances. These classifications are made using the available data.

An important feature of Naive Bayes classifier is that it only requires a small amount of training data to estimate the parameters necessary for classification. For some types of models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting.

In spite of its oversimplified assumptions, Naive Bayes classifiers have worked efficiently in many complex real-world situations. These have worked well in spam filtering and document classification.[6]

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Figure 1.8: Naïve Bayes' Formula

Above,

- $P(c|x)$ is the posterior probability of class (c, target) given predictor (x, attributes).
- $P(c)$ is the prior probability of class.
- $P(x|c)$ is the likelihood which is the probability of predictor given class.
- $P(x)$ is the prior probability of predictor.

Let's understand it using an example. Below I have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let's follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Figure 1.1: Data Set, Frequency Table and Likelihood Table

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Problem: Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

1.3 Summary

Here is the description of the problem identified and the theoritical background. The theoritical concepts involved in the project includes nltk, tweepy and textblob which are the dependencies for the project. Some other python libraries that play a major role in the implementation are matplotlib for ploting histograms, pie charts and graphs the others are CSV, RE for the csv file conversion and regular expression operations. Data preprocessing converts the raw data to clean data. Sentiment analysis is based on classification which in turn is a part of supervised learning.

CHAPTER 2

Software Requirements Specification

2.1 Introduction

The aim of this project is to classify tweets from Twitter into “positive”, “negative” or “neutral”. However, this is done only in the older version. In the newer version, tweets will be further classified into “weakly positive”, “strongly positive”, “weakly negative” and “strongly negative”. Twitter is a microblogging website where people can share their feelings quickly a nd spontaneously by sending a tweets limited by 140 characters. Tweets can be directly addressed by adding the target sign “@” or participate to a topic by adding an hashtag “#” to t he tweet. Because of the usage of Twitter, it is a perfect source of data to determine the curre nt overall opinion about anything.

2.1.1 Purpose

The project focuses on allowing the user to gain an overview of the wider public opinion behind certain topics. Being able to quickly see the sentiment behind everything from forum posts to news articles means being better able to strategise and plan for the future.

2.1.2 Intended Audience and Reading Suggestions

This application aims in helping the organisations to be able to monitor the different social media sites in real time and act accordingly. Marketing managers, PR firms, campaign managers, politicians and even equity investors and online shoppers are the direct beneficiaries of sentiment analysis technology.

2.1.3 Product Scope

The scope of the project is to provide a user friendly web based product that extracts people’s sentiment feelings toward certain services, products, organizations, political or nonpolitical topics and any influential people on social media. The project aims to provide an accurate sentiment analysis result. It provides a plenty of options in terms of filtering and viewing information according to user’s needs.

2.2 Overall Description

It defines the product perspective, product functions and the design and implementation constraints.

2.2.1 Product Perspective

The main aim of the project is to crawl over Twitter collecting all tweets related to the user's search keyword. Then perform an intelligent processing technique to extract the true meanings of the people's comments and to decide and classify them in terms of positive, negative or neutral thus to know the majority of people like or dislike the desired topic. More specifically providing people's feelings regarding certain topics with high accuracy will lead to a better decision making. The purpose of the project is to demonstrate the concept and to deliver operational and functional services for testing purposes.

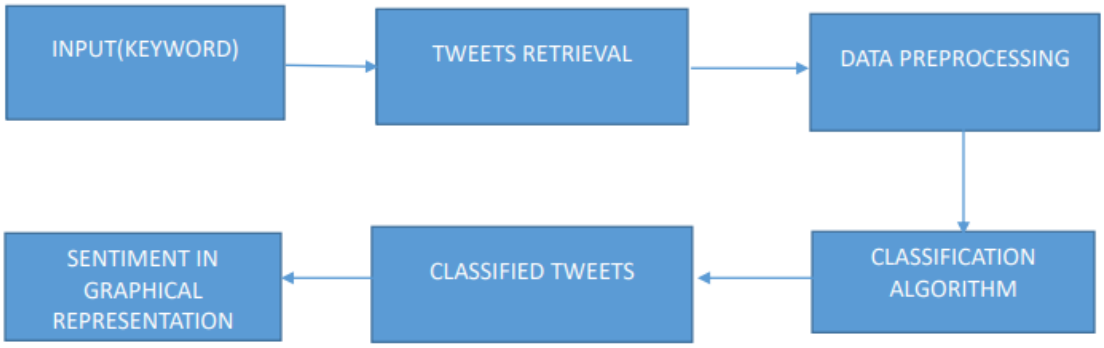

Figure 2.1: Block diagram of the process

2.2.2 Product Functions

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.
- Perform Sentiment Analysis on the tweets stored 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 to that topic.
- Summarize the major functions the product must perform or must let the user perform.

2.2.3 User Classes and Characteristics

The intended user will be a member of the general public who is interested in the sentiment of the Twitter population with respect to various topics. Users are not expected to have a very high level of technical expertise.

2.2.4 Operating Environment

Software requirements:

- Linux Operating System/Windows
- Python Platform (Anaconda2,Spyder,Jupyter)
- NLTK package,
- Modern Web Browser
- Twitter API, Google API

Hardware requirements:

- A working computer system with a minimum of 3 Gb RAM.

2.2.5 Design and Implementation Constraints

The project will be developed keeping in mind the following constraints.

2.2.5.1 Operating System

The project shall be constrained to the Windows and Linux operating system.

2.2.5.2 Graphical User Interface

The project shall be written to be executed on any LINUX/Windows OS. As such the projecct shall not be optimized for any other operating system.

2.2.5.3 Data Output

The output shall be a visual output that will allow the user to see the result of the analysis in graphical form.

2.2.6 Assumptions and Dependencies

The assumptions and dependencies necessary for the development of the project are stated as follows.

2.2.6.1 Sentiment Analysis

An assumption is that it is possible to accurately determine the sentiment for a 140 character string of English text.

2.2.6.2 Internet

Internet access is required for each analysis session to function properly. An assumption is that the device maintains Internet access throughout the entire analysis session.

2.3 External Interface Requirements

This describes the logical and physical characteristics of each interface between the software product and the hardware components of the system.

2.3.1 Hardware Interfaces

For the product to run, a working computer system is needed. To get accurate results, the device should have a stable internet connection. The minimum capacity for the RAM should be of 3 gigabytes for proper functioning of the software resources.

2.3.2 Software Interfaces

As it has already been stated, the product shall only run on Windows /Linux operating system and it requires Anaconda - Python other than these.

2.3.2.1 Inputs

The software will receive input from two sources. First, the user interface and second, the Twitter API. The user interface will supply the keywords and the analysis session duration, while the Twitter API will supply the Tweet text.

2.3.2.2 Outputs

The output will portray the current mood of the Twitter community on a given topic in the form of a simple gauge. If available, historical data will be displayed in a graph.

2.3.2.3 Operating System

The software will run on Windows/ Linux operating system.

2.3.3 Communications Interfaces

As it has already been stated, Internet connection is obviously necessary for a complete operation but, no special configuration for the network adapter is needed.

2.4 Other Non-functional Requirements

Non-functional requirements specify the criteria that can be used to judge the operations of a system.

2.4.1 Performance Requirements

The processing shall not cause other applications to halt/hang or exit prematurely.

2.4.2 Security Requirements

This project does not directly deal with any personal information that requires a secure application it will never ask for personal information, in any form.

2.4.3 Software Quality Attributes

Quality assurance activities are oriented towards prevention of introduction of defects.

2.4.3.1 Availability

This project will be available with a stable internet connection.

2.4.3.2 Maintainability

The project should be developed clearly and concisely. The code will be well documented. Particular care will be taken to design the project modularly to ensure that maintenance is easy.

2.4.3.3 Transferability/Portability

This project is specifically saved to the device it is installed on, and does not give any cloud storage. Upon reinstalling, no settings are guaranteed to be saved.

2.5 Functional Requirements

Functional requirements may involve calculations, technical details, data manipulations and processing and other specific functionality that defines what a system is supposed to accomplish.

2.5.1 Retrieving Input

The software will receive three inputs: keywords, analysis session duration, and Tweets.

- Keywords will be entered by the user for each topic.
- The analysis session duration will be set by the user before each session.
- Tweets will be retrieved with the Twitter Streaming API.

2.5.2 Real-Time Processing

The software will take input, process data, and display output in real-time. This will enforce that the snapshot provided by the simple gauge is a current view of the Twitter community's mood on the chosen topic.

2.5.3 Sentiment Analysis

Sentiment analysis will be performed on the user-specified keywords within the Tweet to determine the overall mood of the Tweet relative to the topic. The sentiment analysis will provide a negative, neutral,or positive numeric sentiment value.

2.5.4 Output

The software must output real time data in the form of a simple gauge. In addition, the software may output a graph of mood trends over time, as well as additional statistics pertaining to a topic (average sentiment over all analysis sessions and total number of tweets processed). This output should be clear and easy to understand.

2.6 Summary

This chapter represents the SRS as a description of a software system to be developed. It lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction. It also establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function. Here, it provides a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign.

The chapter provides the purpose, overall description, design and implementation constraints, interfaces and the requirements needed to conduct the sentiment analysis efficiently.

CHAPTER 3

Design

3.1 Introduction

The design phase is the core phase of the software development process. The system requirements and logical description of the entities, relationships, and attributes of the data that were documented during the Requirements Analysis Phase are further refined and allocated into system and database design specifications that are organized in a way suitable for implementation within the constraints of a physical environment (e.g., computer, database, facilities).

In this chapter, based on the requirements captured in SRS, architecture design is proposed for project and captured in design document. This phase includes:-

- Partition of requirements into hardware & software system.
- Designing system architecture
- Creating UML diagrams (Use cases, class diagram, sequence diagrams and Activity diagram)

3.2 System Flow Diagram

A system flow diagram is a way to show relationships between a business and its components, such as customer. In a system flow diagram, the goal is to present a visual representation of some component of the business model. For this project, the training data is passed through a machine learning algorithm which uses a classifier to predict the sentiment for a new data.

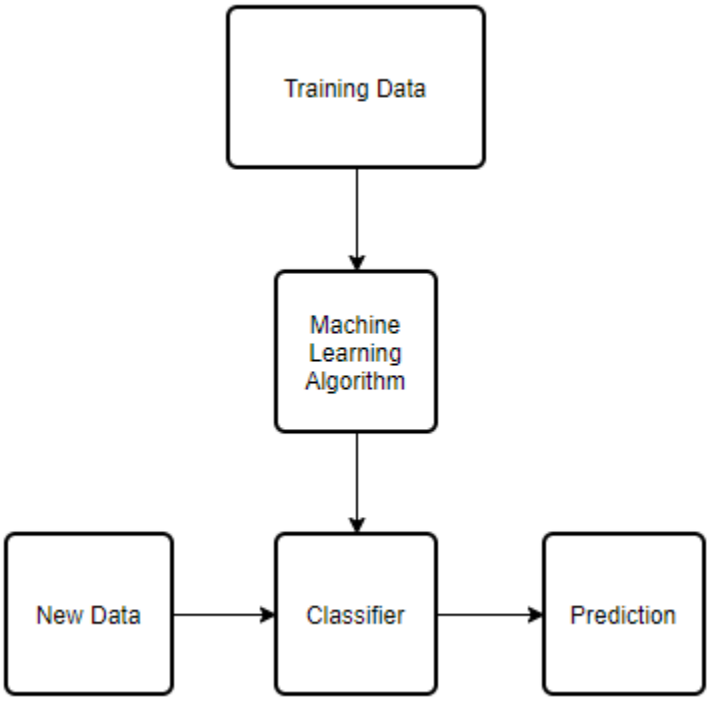

Figure 3.1: System Flow Diagram

3.3 Activity Diagram

The following activity diagram represents the flow from one activity to another. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. In this diagram, the sentiment Analysis procedure is represented through a series of steps. Firstly, the keyword to be searched is asked from the user. The related data is extracted in order to process the tweets. Next, the training data is used for feature extraction and through Naive Bayes' algorithm, the sentiment is predicted. The result is then displayed using a pie chart and a bar graph.

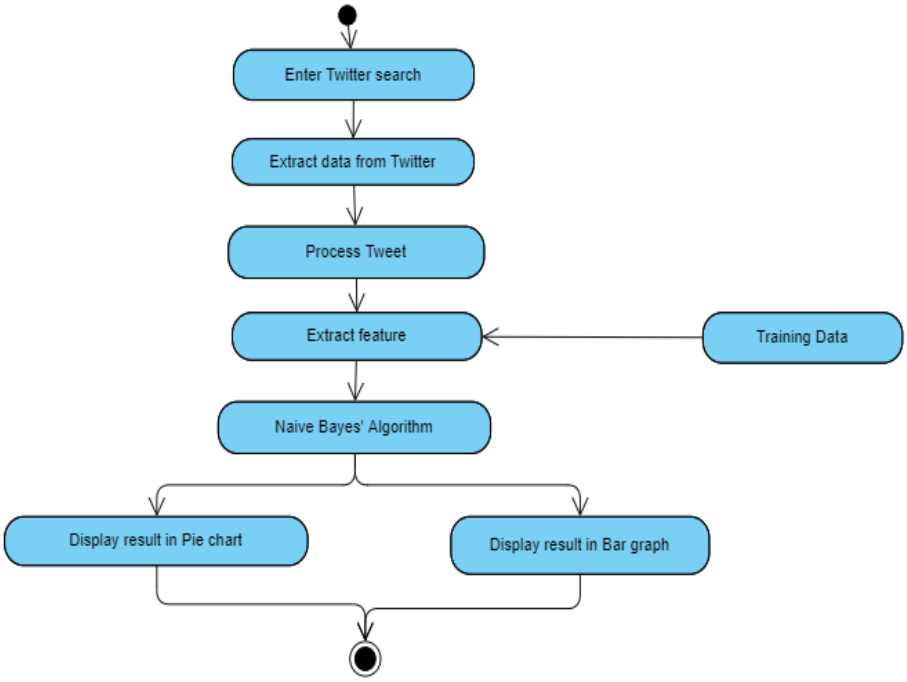

Figure 3.2: Activity Diagram

3.4 Data Flow Diagram

The following diagram shows the graphical representation of the flow of data through this information system, modeling its process aspects. Here, the data flow shows the steps in which, initially, the user provides the keyword to be evaluated. This is followed by extraction and processing of tweets. Feature Extraction is performed next in order to classify the tweets through the Naïve Bayes' Algorithm followed by polarity calculation.

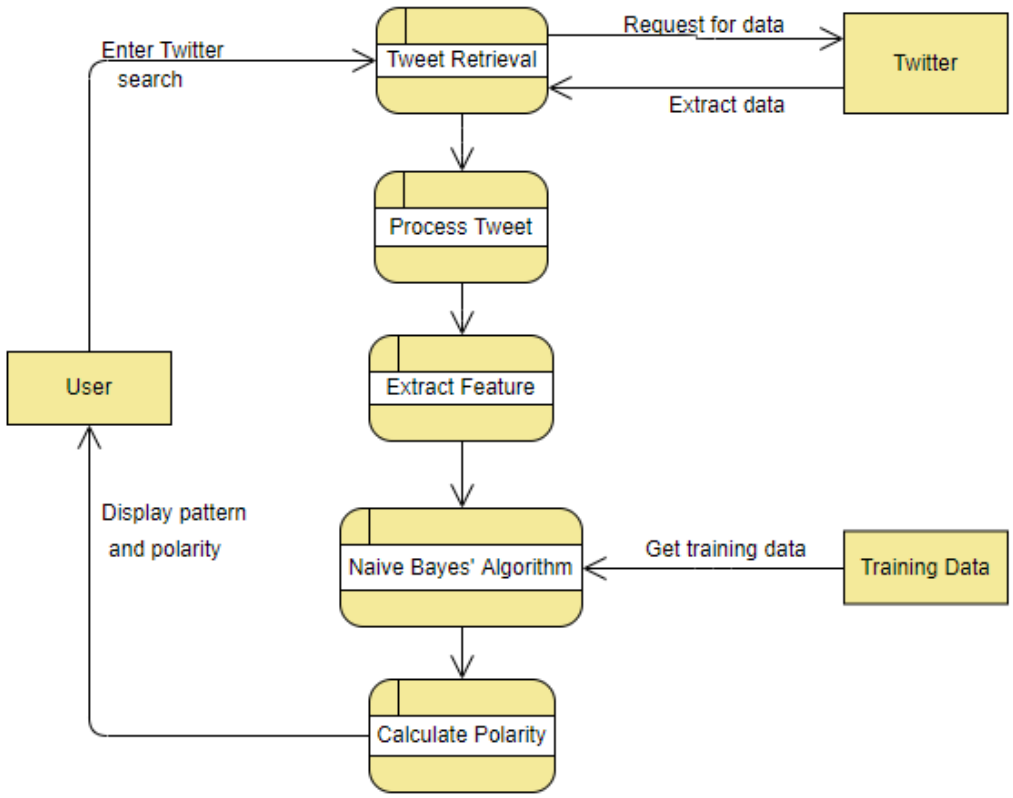

Figure 3.3: Data Flow Diagram

3.5 Sequence Diagram

The following sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. The diagram represents the sequence in which the procedure is executed. The word to be searched is asked from the user, followed by extraction. Then the tweets are classified for sentiment prediction.

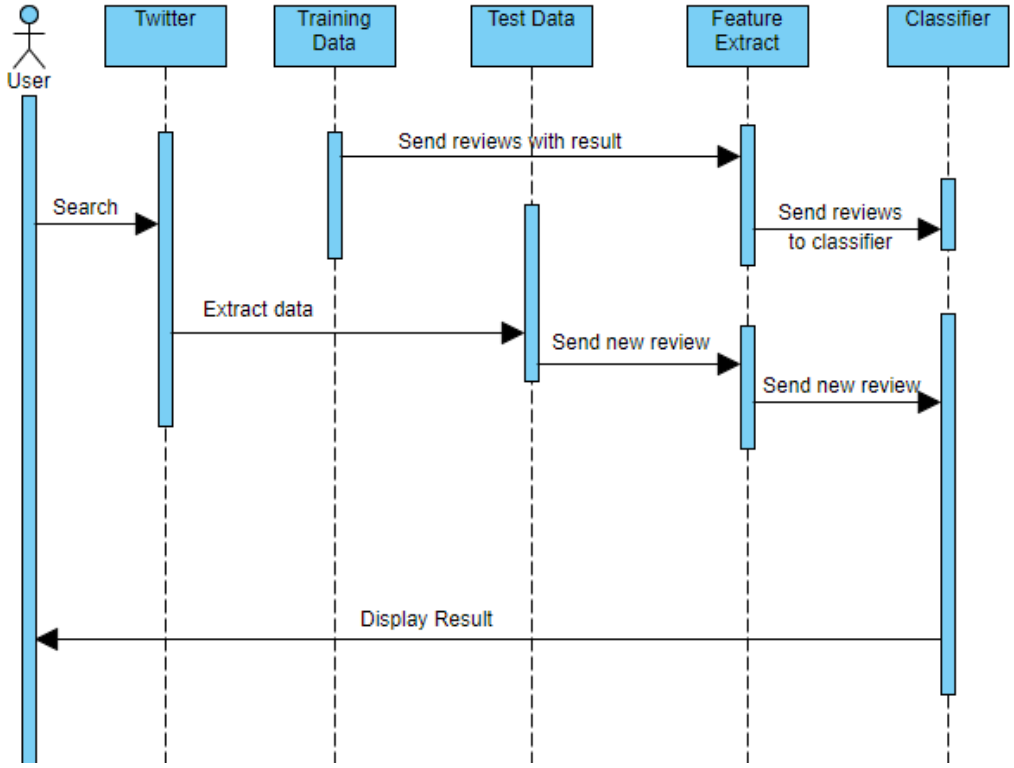

Figure 3.4: Sequence Diagram

3.6 Use Case Diagram

Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. The user and the micro-blogging website Twitter perform the role of actors here that are responsible for the process of tweet retrieval, tweet extraction, tweet classification and finally, sentiment prediction.

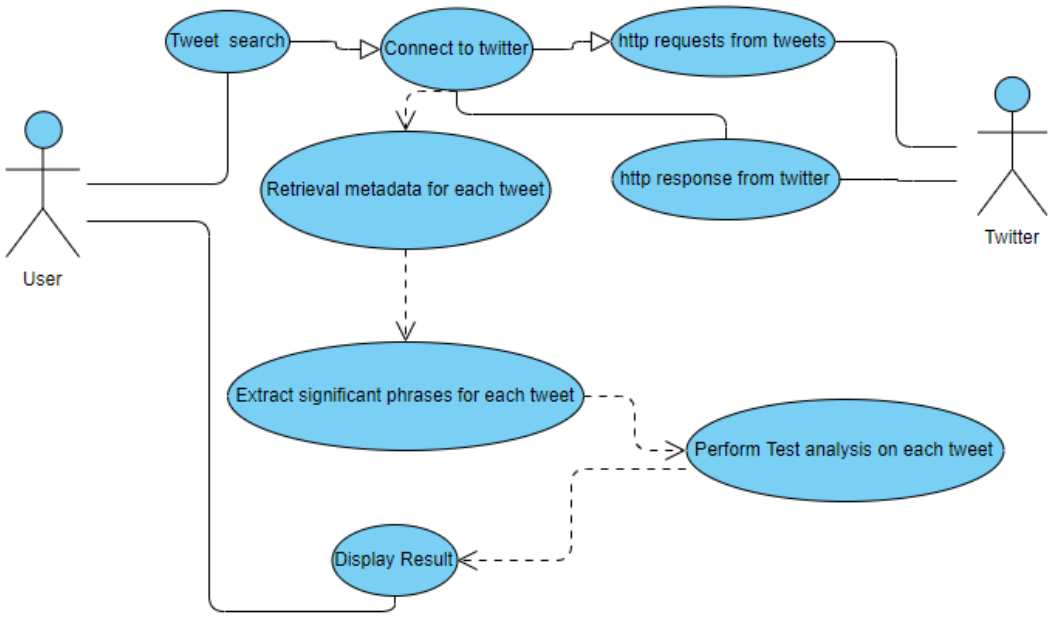

Figure 3.5: Use Case Diagram

3.7 Summary

This chapter aims at the conduction of a formal review of the high-level architectural design to achieve confidence that the design satisfies the system requirements, is in conformance with the architecture and prescribed design standards, to raise and resolve any critical technical and/or project-related issues, and to identify and mitigate project, technical, security, and/or business risks affecting continued detailed design and subsequent lifecycle activities. During the Design Phase, the initial strategy for any necessary training is also begun. In addition, the work planned for future phases is redefined, if necessary, based on information acquired during the Design Phase.

In order to perform proper implementation, UML models are constructed for pictorial representation. Hence, a system flow diagram, activity diagram, data flow diagram, sequence

diagram and a use case diagram have been made that represent the sentiment analysis procedure through a series of steps.

CHAPTER 4

Implementation and Results

4.1 Introduction

The project takes shape during the implementation phase. This chapter involves the construction of the actual result for the Sentiment Analysis. It is the doing phase, and it is important to maintain the momentum. Here the process of tweet extraction is performed through the properties of the NLP basics i.e. tokenization, stop words and text blob. The code for the first version of the software, where the tweets are classified into the positive, negative and neutral category, respectively, is included in the chapter. Similarly, for the second version , where the tweets are classified into positive, weakly positive, strongly positive, negative, weakly negative, strongly negative and neutral category, is included too, along with the results. The various areas where the project might find its applications along with its limitations are listed here, as well.

4.2 Natural Language Processing Basics

- Stopwords: Stop words are words which are filtered out before or after processing of natural language data. The *figure 4.2(a)* and *figure 4.2(b)* showcases stopword filteration in a sentence.

```
C:\Users\Radhika\Desktop\Major>python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> from nltk.corpus import stopwords
>>> from nltk.tokenize import word_tokenize
>>> sent="Radhika, varnika and kaavya are working on sentiment analysis."
>>> stop_words= set(stopwords.words('english'))
>>> print(stop_words)
{"wasn't", 'own', 'which', "don't", 'wouldn', 'should', 'there', "you'll", 'ma', "you've", 'below', 'i', "wouldn't", 'can', 'yours', 'between', 'themselves', 'few', 'th
is', 'as', 'will', 'each', 'me', 'myself', "mustn't", "hadn't", 'has', 'further', 'did', 'no', 'so', "needn't", 'theirs', 'in', 'o', 'our', 'weren', "won't", 'because',
 'very', 'from', 'ourselves', 'does', 'same', "couldn't", 'before', 'm', 'aren', 'do', "isn't", 'my', 'ours', 'where', 'll', 'won', 'out', 'how', 'such', 'during', 'had
n', 'all', 'had', 'was', 'both', 'wasn', 'haven', 'doing', 'y', 'under', 'were', 'him', 'a', 'for', 've', 'by', 'is', 'or', 'until', 'his', 'when', 'herself', 'down', '
that', "you're", 'yourselves', 'against', 'while', "shan't", 're', 'are', 's', 'why', 't', 'any', 'through', 'again', 'once', 'but', 'than', 'd', 'them', 'nor', 'hers',
 'some', 'too', "doesn't", "should've", 'have', 'he', "mightn't", 'into', 'only', 'their', 'ain', 'above', 'now', 'most', 'not', 'on', "it's", 'those', 'the', 'and', 'w
hat', "aren't", 'needn', 'himself', 'to', 'having', 'with', 'after', 'about', 'been', 'itself', 'being', 'more', 'these', 'who', 'didn', 'other', 'she', 'we', 'yourself
', "that'll", 'be', 'couldn', "she's", 'mightn', 'its', "weren't", 'they', 'you', 'mustn', 'if', 'it', 'her', 'an', 'off', "shouldn't", 'then', 'hasn', 'up', 'am', 'you
'd', 'here', 'whom', "haven't", 'at', 'don', 'just', 'isn', 'shouldn', 'of', 'your', 'over', 'doesn', "hasn't", 'shan', "didn't"}
>>>
```

Figure 4.1(a): Stopwords for the English language.

```
>>> words=word_tokenize(sent)
>>> filtered_sent=[]
>>> for i in words:
...     if i not in stop_words:
...             filtered_sent.append(i)
...
>>> print(filtered_sent)
['Radhika', ',', 'varnika', 'kaavya', 'working', 'sentiment', 'analysis', '.']
>>>
```

Figure 4.1(b): Stopwords removal.

- Tokenization: Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens. Tokens can be

individual words, phrases or even whole sentences. In the process of tokenization, some characters like punctuation marks are discarded as shown in figure 4.1(c).

```
>>> import nltk
>>> from nltk.tokenize import word_tokenize, sent_tokenize
>>> sent= "Hello Radhika. How are you doing today? "
>>> print(sent_tokenize(sent))
['Hello Radhika.', 'How are you doing today?']
>>> print(word_tokenize(sent))
['Hello', 'Radhika', '.', 'How', 'are', 'you', 'doing', 'today', '?']
>>>
```

Figure 4.1(c): Tokenization

4.3 Textblob

Textblob: It is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation.

```
>>> from textblob import TextBlob
>>> a=TextBlob("Radhika is the worst programmar")
>>> a.sentiment.polarity
-1.0
>>> a=TextBlob("Radhika is the best programmar")
>>> a.sentiment.polarity
1.0
>>> a=TextBlob("Radhika is a programmar")
>>> a.sentiment.polarity
0.0
```

Figure 4.2: Textblob processing textual data

4.4 Version 1.0

Here is the implementation of twitter sentiment analysis using tweepy and textblob and the tweets are categorized in positive and negative, sentiment value is displayed in the form of percentage and the resulting tweets which are positive and negative are also displayed.

```
C:\Users\Radhika\Desktop\Major>python perc.py
Positive tweets percentage: 21.311475409836067 %
Negative tweets percentage: 24.59016393442623 %


Positive tweets:
@AnnCoulter Ann needs to spend more time talking about the Candidate &amp;amp; the Truth about Donald Trump. You have label… https://t.co/MI6TZtUu1B
RT @AdamSchiff: Misrepresenting the Special Counsel investigation, attempting to exonerate a president who may have committed multiple felo…
RT @ChuckCallesto: Raise you hand if you think Donald Trump's Pennsylvania RALLY will have more OVERFLOW attendees than all 2020 DEM candid…
RT @WildPalmsLtd: Not "unsettled" enough to act. None of these Republicans will really feel the pain until there's a Democratic President.…
TICKET: Donald Trump, Butt touching water fountain, $788
RT @SkyNewsAust: An exodus of Australians bound for New Zealand could be underway in the wake of the Coalition’s re-election on Saturday, s…
RT @JohnFugelsang: America's federal ban on female genital mutilation is about to lapse.
Donald Trump's solicitor general has confirmed tha…
Living in an alternative Universe. The very real dangers of Donald Trump. https://t.co/lfIKPbbGpK
@ACTBrigitte This is a parody account, right? No one could pack that much ignorance and inaccuracy in a single twee… https://t.co/SrX0fKPSah
RT @EdwardTHardy: Richard Nixon's Attorney General John Mitchell spent 19 months in prison for doing what Donald Trump's Attorney General W…


Negative tweets:
RT @gtconway3d: “‘I felt the rules were being changed to hurt Trump, and I thought it was damaging for the presidency over the long haul,’…
@Hessian_Mohd That is, in fact, why Donald Trump works SO HARD to convince his supporters to be uninformed:

if the… https://t.co/KGQLXSd6lJ
#Amash is the only GOP member with a pair of balls!!  Perhaps one of the few members of Congress with any!!
Presid… https://t.co/Qci1XM90gc
RT @johnpavlovitz: "Donald Trump showed you who he was.
As ugly as he is—he comes as advertised.
He prophesied this full-blown kleptocra…
RT @EricLiptonNYT: JUST POSTED: He is a Canadian citizen. A billionaire. And has been lobbying the Trump administration hard to promote ste…
Iranian Official Calls Donald Trump "Crazy President" https://t.co/noVqadW8ZQ
RT @funder: "Oh my God. This is terrible. This is the end of my Presidency. I’m fucked. How could you let this happen, Jeff? This is the wo…
RT @JeannePancurak: @tedcruz "Knock the crap out of them, would you? Seriously, OK? Just knock the  hell, I promise you I will pay for the…
RT @StevenBeschloss: With our United States of America plane behind him, the lying demagogue Donald Trump fuels the crowd to shout hostilit…
Don McGahn has bigger problems than deciding whether to follow Donald Trump’s illegal order not to testify… https://t.co/W2Hlu2eCoU
```

Figure 4.3: Resulting tweets and their percentage of polarity

4.5 Version 1.1

The implementation results for the older version are displayed where the tweets are classified into positive, negative and neutral.

Implementation 1: Sentiment Analysis for the word ‘bitcoin’.

```
C:\Users\Radhika\Desktop\Major>python old.py
Enter Keyword/Tag to search about: bitcoin
Enter how many tweets to search: 180
How people are reacting on bitcoin  by analyzing 180 Tweets.
Positive
```

Figure 4.4(a): 'bitcoin' evaluated among number of tweets

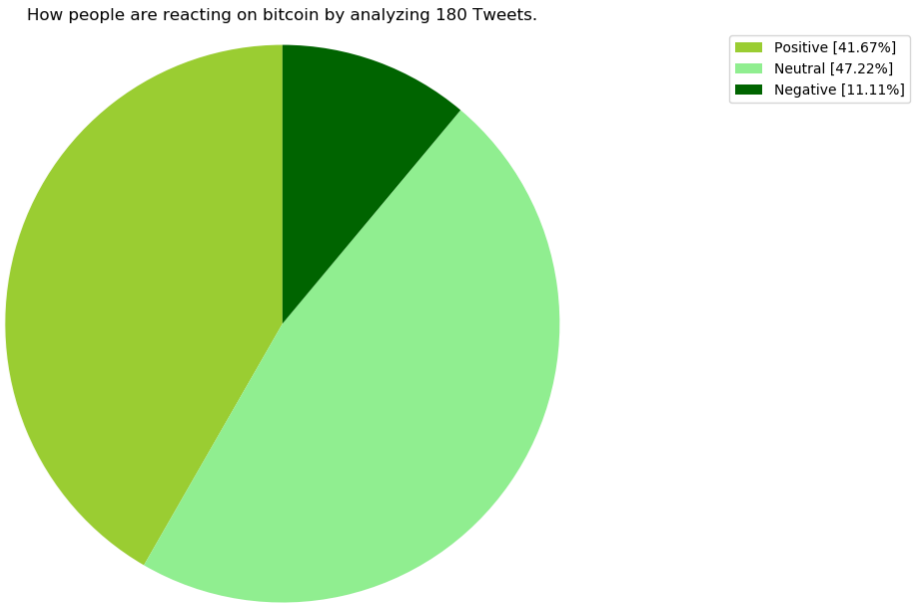

Figure 4.4(b): Evaluation result

Implementation 2: Sentiment Analysis for the word ‘rafale’.

```
C:\Users\Radhika\Desktop\Major>python old.py
Enter Keyword/Tag to search about: rafale
Enter how many tweets to search: 250
How people are reacting on rafale by analyzing 250 Tweets.
Positive
```

Figure 4.4(c): 'rafale' evaluated among number of tweets.

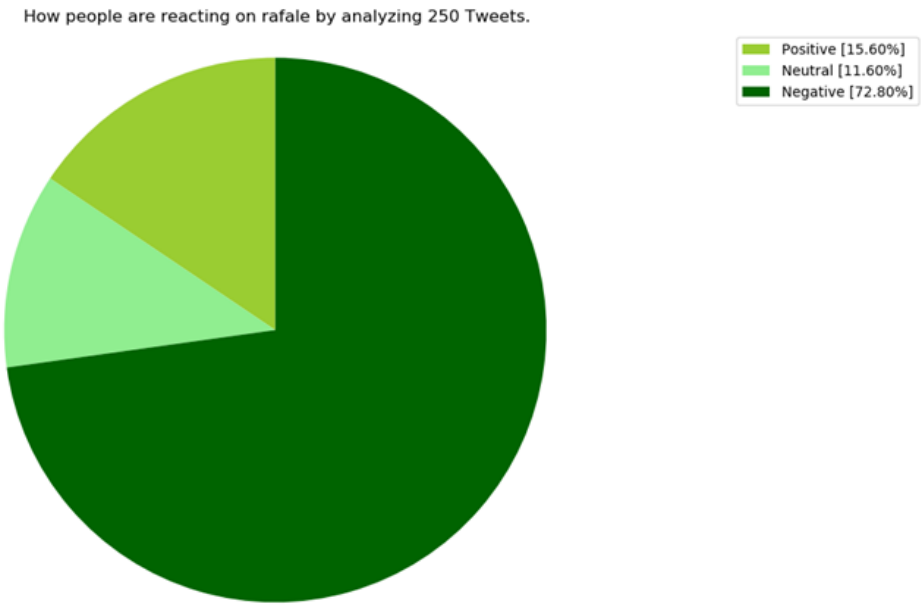

Figure 4.4(d): Evaluation result.

Implementation 3: Sentiment Analysis for the word ‘Narendra Modi’.

```
C:\Users\Radhika\Desktop\Major>python old.py
Enter Keyword/Tag to search about: narendra modi
Enter how many tweets to search: 200
How people are reacting on narendra modi by analyzing 200 Tweets.
Positive
```

Figure 4.4(e): 'narendra modi' evaluated among number of tweets.

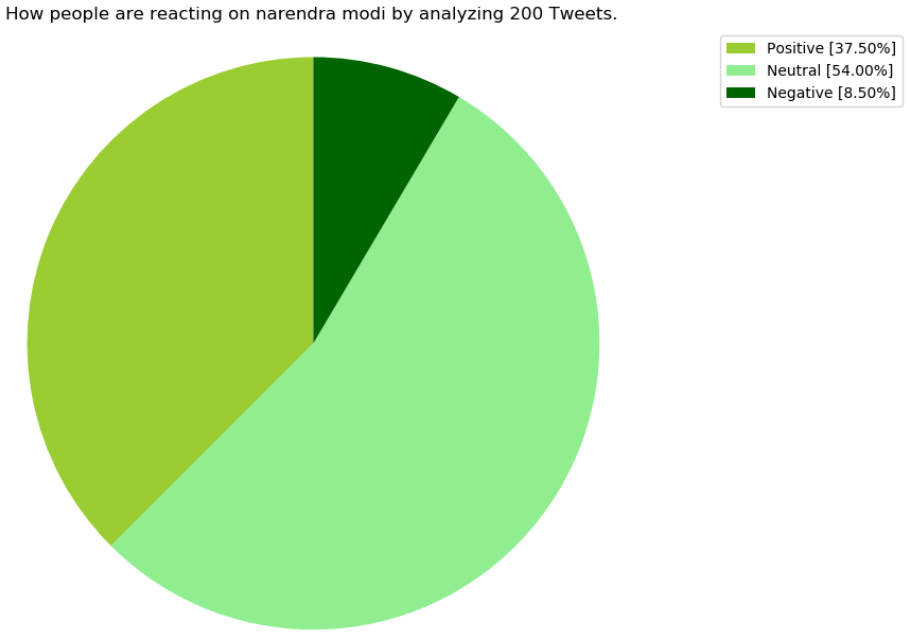

Figure 4.4(f): Evaluation result.

4.6 Version 1.2

The implementation results for the newer version are displayed where the tweets are further classified into strongly positive, weakly positive, strongly negative and weakly negative.

Implementation 1: Sentiment Analysis for the word ‘bitcoin’.

```
C:\Users\Radhika\Desktop\Major>python neww.py
Enter Keyword/Tag to search about: bitcoin
Enter how many tweets to search: 180
How people are reacting on bitcoin  by analyzing 180 tweets.

General Report:
Weakly Positive
```

Figure 4.5(a): 'bitcoin' evaluated among number of tweets.

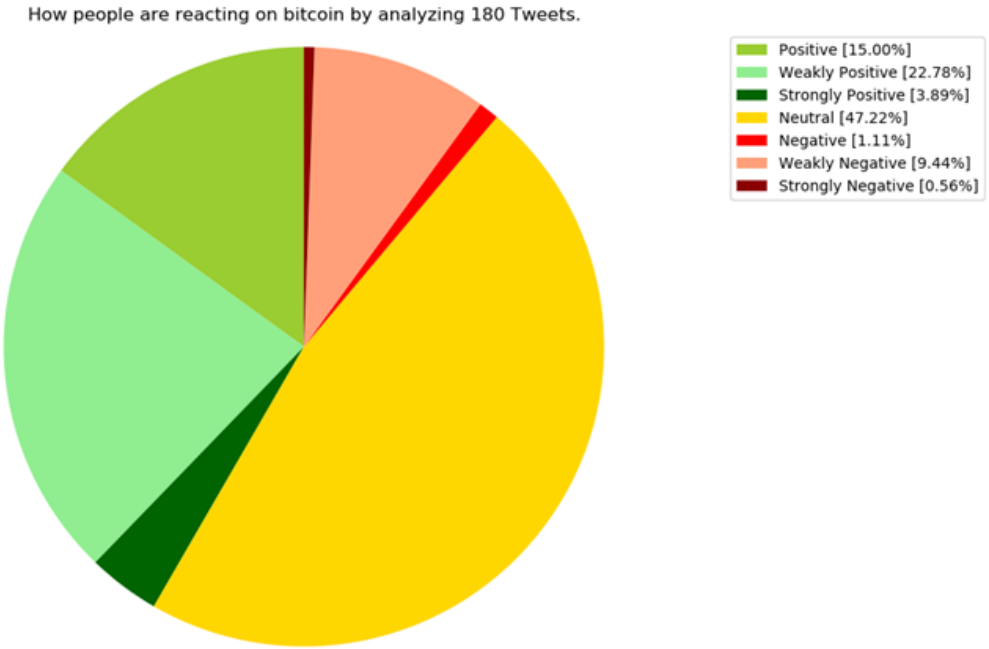

Figure 4.5(b): Evaluation result.

Implementation 2: Sentiment Analysis for the word ‘rafale’.

```
C:\Users\Radhika\Desktop\Major>python neww.py
Enter Keyword/Tag to search about: rafale
Enter how many tweets to search: 250
How people are reacting on rafale by analyzing 250 tweets.

General Report:
Weakly Positive
```

Figure 4.5(c): 'rafale' evaluated among number of tweets.

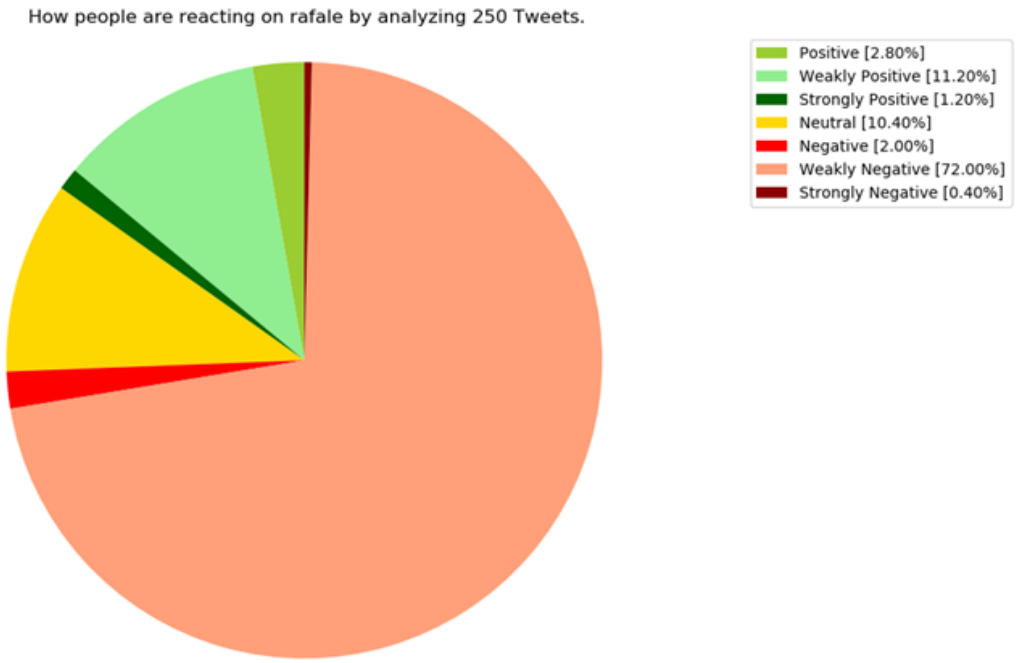

Figure 4.5(d): Evaluation result.

Implementation 3: Sentiment Analysis for the word ‘Narendra Modi’.

```
C:\Users\Radhika\Desktop\Major>python neww.py
Enter Keyword/Tag to search about: narendra modi
Enter how many tweets to search: 200
How people are reacting on narendra modi by analyzing 200 tweets.

General Report:
Weakly Positive
```

Figure 4.5(e): 'narendra modi' is evaluated among number of tweets.

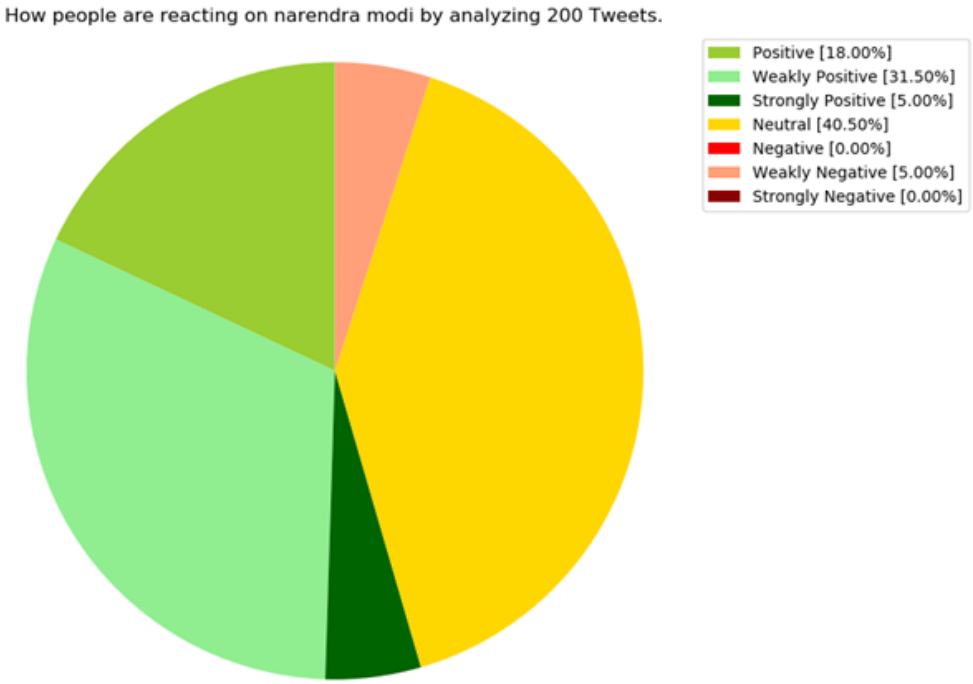

Figure 4.5(f): Evaluation result.

4.7 Scope of the Project

Sentiment analysis has many applications and benefits to your business and organization.

1. It can be used to give your business valuable insights into how people feel about your product brand or service.
2. When applied to social media channels, it can be used to identify spikes in sentiment, thereby allowing you to identify potential product advocates or social media influencers.
3. It can be used to identify when potential negative threads are emerging online regarding your business, thereby allowing you to be proactive in dealing with it more quickly.
4. Sentiment analysis could also be applied to your corporate network, for example, by applying it to your email server, emails could be monitored for their general“tone”.

4.8 Limitations

1. Computer programs have problems recognizing things like sarcasm and irony, negations, jokes, and exaggerations - the sorts of things a person would have little trouble identifying. And failing to recognize these can skew the results.
2. 'Disappointed' may be classified as a negative word for the purposes of sentiment analysis, but within the phrase “I wasn't disappointed", it should be classified as positive.
3. We would find it easy to recognize as sarcasm the statement "I'm really loving the enormous pool at my hotel!", if this statement is accompanied by a photo of a tiny swimming pool; whereas an automated sentiment analysis tool probably would not, and would most likely classify it as an example of positive sentiment.
4. With short sentences and pieces of text, for example like those you find on Twitter especially, and sometimes on Facebook, there might not be enough context for a reliable sentiment analysis. However, in general, Twitter has a reputation for being a good source of information for sentiment analysis, and with the new increased word count for tweets it's likely it will become even more useful.

So, automated sentiment analysis tools do a really great job of analysing text for opinion and attitude, but they're not perfect.

4.9 Summary

This chapter includes all the implementations of the three versions. Natural language processing is all about processing the language with the help of different processes such as word tokenization and sentence tokenization, stop words elimination etc.Version 1.0 is all about the percentages of the sentiment polarity of negative and positive categories and then 5 tweets of each category is displayed. Version 1.1 is all about the pie charts of the resulting values.Version 1.2 has a further bifurcation of positive and negative categories into strongly positive, weakly positive and strongly negative, weakly negative respectively.

REFERENCES

[1] https://pdfs.semanticscholar.org/c114/7f3d9b46ff0a0c7c43b668123cb15a26120d.pdf

[2]https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_data_preprocessing_analysis_visualization.htm

[3]https://globallogic.com/wp-content/uploads/2014/10/Introduction-to-Sentiment-Analysis.pdf

[4]https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_ training_test_data.htm

[5]https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_types_of_learning.htm

[6]https://www.tutorialspoint.com/data_mining/dm_bayesian_classification.htm

[7]https://www.pythoncentral.io/introduction-to-tweepy-twitter-for-python/

[8]https://planspace.org/20150607-textblob_sentiment/

[9]https://www.google.com/amp/s/www.geeksforgeeks.org/python-introduction-matplotlib/amp/

[10]https://realpython.com/python-csv/

[11]https://www.programiz.com/python-programming/regex

[12]https://www.python-course.eu/sys_module.php

APPENDIX

Version 1.0

```
import re
import tweepy
from tweepy import OAuthHandler
from textblob import TextBlob

class TwitterClient(object):

    def __init__(self):

        consumer_key = 'gUMr9AbzAH7oaPt52cRppoSHG'
        consumer_secret = 'JOofvs0CiZd09GQl8y9Bflip4hL2SVZy0Z86tDpW7yKQ8O2ENF'
        access_token = '1108107688800477184-CZSfsoOU24rXl22fMyFbyfHw2PpLZx'
        access_token_secret = '5zREPBzLSvKEA3p43JbJoY9IvvC5Sb5LqNqzrW4Nbc5sA'
        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 = 'Donald Trump', 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("\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()
```

Version 1.1

```
from textblob import TextBlob
import sys, tweepy
import matplotlib.pyplot as plt


def percentage(part, whole):
   return 100*float(part)/float(whole)


consumerKey = "gUMr9AbzAH7oaPt52cRppoSHG"
consumerSecret = "JOofvs0CiZd09GQl8y9Bflip4hL2SVZy0Z86tDpW7yKQ8O2ENF"
accessToken = "1108107688800477184-CZSfsoOU24rXl22fMyFbyfHw2PpLZx"
accessTokenSecret = "5zREPBzLSvKEA3p43JbJoY9IvvC5Sb5LqNqzrW4Nbc5sA"

auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
auth.set_access_token(accessToken, accessTokenSecret)
api = tweepy.API(auth)

searchTerm = input("Enter Keyword/Tag to search about: ")
NoOfTerms = int(input("Enter how many tweets to search: "))

tweets = tweepy.Cursor(api.search, q=searchTerm, lang = "en").items(NoOfTerms)

polarity = 0
positive = 0
negative = 0
neutral = 0

for tweet in tweets:
   analysis = TextBlob(tweet.text)
   polarity +=analysis.sentiment.polarity

   if (analysis.sentiment.polarity == 0):
      neutral += 1
   elif (analysis.sentiment.polarity < 0):
      negative +=1
   elif (analysis.sentiment.polarity > 0):
      positive +=1

positive = percentage(positive, NoOfTerms)
negative = percentage(negative, NoOfTerms)
neutral = percentage(neutral, NoOfTerms)

positive = format(positive, '.2f')
negative = format(negative, '.2f')
neutral = format(neutral, '.2f')

print("How people are reacting on " + searchTerm + " by analyzing " + str(NoOfTerms) + "
Tweets.")

if (polarity == 0):
   print('Neutral')
elif (polarity < 0):
   print ('Negative')
elif (polarity > 0):
   print ('Positive')
```

```
labels = ['Positive [' + str(positive) + '%]', 'Neutral [' + str(neutral) + '%]','Negative [' + str(negative) + '%]']
sizes = [positive, neutral, negative]
colors = ['yellowgreen','lightgreen','darkgreen']
patches, texts = plt.pie(sizes, colors=colors, startangle=90)
plt.legend(patches, labels, loc="best")
plt.title('How people are reacting on ' + searchTerm + ' by analyzing ' + str(NoOfTerms) + ' Tweets.')
plt.axis('equal')
plt.tight_layout()
plt.show()
```

Version 1.2

```
import sys,tweepy,csv,re
from textblob import TextBlob
import matplotlib.pyplot as plt


class SentimentAnalysis:

    def __init__(self):
        self.tweets = []
        self.tweetText = []

    def DownloadData(self):
        consumerKey = 'gUMr9AbzAH7oaPt52cRppoSHG'
        consumerSecret = 'JOofvs0CiZd09GQl8y9Bflip4hL2SVZy0Z86tDpW7yKQ8O2ENF'
        accessToken = '1108107688800477184-CZSfsoOU24rXl22fMyFbyfHw2PpLZx'
        accessTokenSecret = '5zREPBzLSvKEA3p43JbJoY9IvvC5Sb5LqNqzrW4Nbc5sA'
        auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
        auth.set_access_token(accessToken, accessTokenSecret)
        api = tweepy.API(auth)

        searchTerm = input("Enter Keyword/Tag to search about: ")
        NoOfTerms = int(input("Enter how many tweets to search: "))

        self.tweets = tweepy.Cursor(api.search, q=searchTerm, lang = "en").items(NoOfTerms)
        csvFile = open('result.csv', 'a')
        csvWriter = csv.writer(csvFile)

        polarity = 0
        positive = 0
        wpositive = 0
        spositive = 0
        negative = 0
        wnegative = 0
        snegative = 0
        neutral = 0

        for tweet in self.tweets:
            self.tweetText.append(self.cleanTweet(tweet.text).encode('utf-8'))

            analysis = TextBlob(tweet.text)
            polarity += analysis.sentiment.polarity
            if (analysis.sentiment.polarity == 0):
                neutral += 1
            elif (analysis.sentiment.polarity > 0 and analysis.sentiment.polarity <= 0.3):
                wpositive += 1
            elif (analysis.sentiment.polarity > 0.3 and analysis.sentiment.polarity <= 0.6):
                positive += 1
            elif (analysis.sentiment.polarity > 0.6 and analysis.sentiment.polarity <= 1):
                spositive += 1
            elif (analysis.sentiment.polarity > -0.3 and analysis.sentiment.polarity <= 0):
                wnegative += 1
            elif (analysis.sentiment.polarity > -0.6 and analysis.sentiment.polarity <= -0.3):
                negative += 1
            elif (analysis.sentiment.polarity > -1 and analysis.sentiment.polarity <= -0.6):
                snegative += 1
```

```
        csvWriter.writerow(self.tweetText)
        csvFile.close()

        positive = self.percentage(positive, NoOfTerms)
        wpositive = self.percentage(wpositive, NoOfTerms)
        spositive = self.percentage(spositive, NoOfTerms)
        negative = self.percentage(negative, NoOfTerms)
        wnegative = self.percentage(wnegative, NoOfTerms)
        snegative = self.percentage(snegative, NoOfTerms)
        neutral = self.percentage(neutral, NoOfTerms)


        polarity = polarity / NoOfTerms

        print("How people are reacting on " + searchTerm + " by analyzing " + str(NoOfTerms)
+ " tweets.")
        print()
        print("General Report: ")


        if (polarity == 0):
            print("Neutral")
        elif (polarity > 0 and polarity <= 0.3):
            print("Weakly Positive")
        elif (polarity > 0.3 and polarity <= 0.6):
            print("Positive")
        elif (polarity > 0.6 and polarity <= 1):
            print("Strongly Positive")
        elif (polarity > -0.3 and polarity <= 0):
            print("Weakly Negative")
        elif (polarity > -0.6 and polarity <= -0.3):
            print("Negative")
        elif (polarity > -1 and polarity <= -0.6):
            print("Strongly Negative")


        self.plotPieChart(positive, wpositive, spositive, negative, wnegative, snegative, neutral,
searchTerm, NoOfTerms)


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

    def percentage(self, part, whole):
        temp = 100 * float(part) / float(whole)
        return format(temp, '.2f')

    def plotPieChart(self, positive, wpositive, spositive, negative, wnegative, snegative, neutral,
searchTerm, noOfSearchTerms):
        labels = ['Positive [' + str(positive) + '%]', 'Weakly Positive [' + str(wpositive) +
'%]','Strongly Positive [' + str(spositive) + '%]', 'Neutral [' + str(neutral) + '%]','Negative [' +
str(negative) + '%]', 'Weakly Negative [' + str(wnegative) + '%]', 'Strongly Negative [' +
str(snegative) + '%]']
        sizes = [positive, wpositive, spositive, neutral, negative, wnegative, snegative]
        colors = ['yellowgreen','lightgreen','darkgreen', 'gold', 'red','lightsalmon','darkred']
        patches, texts = plt.pie(sizes, colors=colors, startangle=90)
        plt.legend(patches, labels, loc="best")
```

```
        plt.title('How people are reacting on ' + searchTerm + ' by analyzing ' + str(noOfSearchTerms) + ' Tweets.')
        plt.axis('equal')
        plt.tight_layout()
        plt.show()


if __name__== "__main__":
    sa = SentimentAnalysis()
    sa.DownloadData()
```