

# **Sentiment Analysis of Movie Data Set**

## **Mini Project Report**

Submitted in partial fulfillment of the requirements for the degree of

### **Bachelor of Engineering (Computer Engineering)**

by:

Danish Khan

TU3S1819022

Shubham Agarwal

TU3F1718124

Jitendra Yadav

TU3S1819007

**Under the Guidance of**

**Prof. Surekha Janrao**



**Department of Computer Engineering**

**TERNA ENGINEERING COLLEGE**

Nerul (W), Navi Mumbai 400706

**(University of Mumbai)**

(2019-2020)

**Internal Approval Sheet**



**TERNA ENGINEERING COLLEGE, NERUL**

**Department of Computer Engineering**

Academic Year 2019-20

**CERTIFICATE**

This is to certify that the major project entitled **“Sentiment Analysis of Movie Data Set”** is a bonafide work of

Danish Khan

TU3S1819022

Shubham Agarwal

TU3F1718124

Jitendra Yadav

TU3S1819007

Submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the Bachelor of Engineering (Computer Engineering).

**Guide**

**Head of Department**

**Principal**

Approval Sheet

**Project Report Approval**

This Major Project Report – an entitled “**Sentiment Analysis of Movie Data Set**” by following students is approved for Mini Project of *Third Year in "Computer Engineering"*.

**Submitted by:**

Danish Khan

TU3S1819022

Shubham Agarwal

TU3F1718124

Jitendra Yadav

TU3S1819007

Examiners Name & Signature:

1. \_\_\_\_\_ - \_\_\_\_\_

2. \_\_\_\_\_ - \_\_\_\_\_

Date: \_\_\_\_\_

Place: \_\_\_\_\_

## **Declaration**

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Danish Khan

TU3S1819022

Shubham Agarwal

TU3F1718124

Jitendra Yadav

TU3S1819007

Date: \_\_\_\_\_

Place: \_\_\_\_\_

## Acknowledgement

We would like to express our sincere gratitude towards our guide **Prof. Surekha Janrao**, guidance and encouragement, they provided during the project development. This work would have not been possible without their valuable time, patience and motivation. We thank them for making my stint thoroughly pleasant and enriching. It was great learning and an honor being their student.

We are deeply thankful to **Dr. Archana Mire (H.O.D Computer Department)** and entire team in the Computer Department. They supported us with scientific guidance, advice and encouragement, they were always helpful and enthusiastic and this inspired us in our work.

We take the privilege to express our sincere thanks to **Dr. L. K. Ragha** our Principal for providing the encouragement and much support throughout our work.

Danish Khan

TU3S1819022

Shubham Agarwal

TU3F1718124

Jitendra Yadav

TU3S1819007

Date: \_\_\_\_\_

Place: \_\_\_\_\_

## **Abstract**

Sentiment analysis or opinion mining is the computational study of people's opinions, sentiments, attributes and emotions expressed in written language. It is one of the most active research area in natural language processing and text mining in recent years. Its popularity is mainly due to two reasons. First, it has a wide range of applications because opinions are central to almost all human activities and are key influencers of our behaviors. Whenever we need to make decisions, we want to hear other's opinions second its presents many challenging research problems, which had never been attempted before the year 2000. Part of the reason for the lack of study before was that there was little opinionated text in digital forms. It is thus no surprise that the inception and the rapid growth of the field coincide with those of the social media on the Web. In fact, the research has also spread outside of computer science to management sciences and social sciences due to its importance to business and society as a whole. In this talk, I will start with the discussion of the mainstream sentiment analysis research and then move on to describe some recent work on modeling comments, discussions, and debates, which represents another kind of analysis of sentiments and opinions. Sentiment classification is a way to analyze the subjective information in the text and then mine the opinion. Sentiment analysis is the procedure by which information is extracted from the opinions, appraisals and emotions of people in regards to entities, events and their attributes. In decision making, the opinions of others have a significant effect on customers ease, making choices with regards to online shopping, choosing events, products, entities. The approaches of text sentiment analysis typically work at a particular level like phrase, sentence or document level. This paper aims at analyzing a solution for the sentiment classification at a fine-grained level, namely the sentence level in which polarity of the sentence can be given by three categories as positive, negative and neutral.

## Table of Contents

	<b>Abstract</b>	i
	<b>List of Figures</b>	ii
	<b>List of Tables</b>	iv
	<b>List of Abbreviations</b>	v
<b>Chapter 1</b>	<b>Introduction</b>	
	1.1 Aim and Objectives of Project	01
	1.2 Scope	02
<b>Chapter 2</b>	<b>Literature Survey</b>	
	2.1 Existing System	03
<b>Chapter 3</b>	<b>Requirement Analysis</b>	
	3.1 SDLC Model Used(Waterfall Model) 3.1.1 Phases Of SDLC Model	05
	3.2 Hardware And Software Requirements	08
<b>Chapter 4</b>	<b>Project Scheduling</b>	
	4.1 Gantt Chart	09
	4.2 Critical Path Method (CPM)	10
	4.3 Earned Value Analysis	11
<b>Chapter 5</b>	<b>Requirement Modeling</b>	

	<b>5.1 Object Oriented Requirement Modeling</b> 5.1.1 Use Case Diagram 5.1.2 Class Diagram	14
	<b>5.2 Structured Requirement Modeling</b> 3.2.1 Data Flow Diagram	16
<b>Chapter 6</b>	<b>Methodology</b>	
	6.1 Project Modules	18
	6.2 Library used	23
<b>Chapter 7</b>	<b>Implementation</b>	
	7.1 Library used	26
<b>Chapter 8</b>	<b>Results And Discussions</b>	
	<b>8.1</b> System Program	31
	8.2 Working Of The System (Screen Snapshots)	34
<b>Chapter 9</b>	<b>Conclusion</b>	35
<b>References</b>		36



## List of Figures

Sr. No	Figure Name	Pg. No
3.1	Waterfall Model	06
4.1	Gantt Chart	09
4.2	Critical Path	11
5.1.1	Use Case Diagram	14
5.1.2	Class Diagram	15
5.2.1	Level 0 and Level 1 DFD Diagram	16
5.2.2	Level 2 DFD Diagram	17
6.1	Implementation Architecture using NLP Approach	24
8.1	Result	34

## LIST OF TABLES

SR.NO	TABLE NAME	PG. NO
4.1	Critical Path Method Table	10

## CHAPTER - 1

### Introduction

We are developing Movie review analyzer by application of NLP, Text Analysis and ML. Sentiment analysis is also popularly known as opinion analysis or opinion mining. The key idea is to use techniques from text analytics, NLP, Machine Learning, and linguistics to extract important information or data points from unstructured text. This in turn can help us derive qualitative outputs like the overall sentiment being on a positive, neutral, or negative scale and quantitative outputs like the sentiment polarity, subjectivity, and objectivity proportions.

Sentiment polarity is typically a numeric score that's assigned to both the positive and negative aspects of a text document based on subjective parameters like specific words and phrases expressing feelings and emotion. Neutral sentiment typically has 0 polarity since it does not express any specific sentiment, positive sentiment will have polarity  $> 0$ , and negative  $< 0$ . Of course, you can always change these thresholds based on the type of text you are dealing with; there are no hard constraints on this

#### 1.1 Aim and Objectives of Project

The main objective in this Project is to predict the sentiment for a number of movie reviews obtained from the Internet Movie Database (IMDb). This dataset contains 50,000 movie reviews that have been pre-labeled with “positive” and “negative” sentiment class labels based on the review content. Besides this, there are additional movie reviews that are unlabeled.

The dataset can be obtained for positive review and negative review from the link below

[https://pythonprogramming.net/static/downloads/short\\_reviews/positive.txt](https://pythonprogramming.net/static/downloads/short_reviews/positive.txt)

[https://pythonprogramming.net/static/downloads/short\\_reviews/negative.txt](https://pythonprogramming.net/static/downloads/short_reviews/negative.txt)

Courtesy of Stanford University and Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. They have datasets in the form of raw text as well as already processed bag of words formats. We will only be using the raw labeled movie reviews for our analyses.

### 1.2 Scope

The scope of the project is sentiment analysis or opinion mining, where we want to analyze some textual documents and predict their sentiment or opinion based on the content of these documents. Sentiment analysis is perhaps one of the most popular applications of natural language processing and text analytics with a vast number of websites, books and tutorials on this subject. Typically sentiment analysis seems to work best on subjective text, where people express opinions, feelings, and their mood. From a real-world industry standpoint, sentiment analysis is widely used to analyze corporate surveys, feedback surveys, social media data, and reviews for movies, places, commodities, and many more. The idea is to analyze and understand the reactions of people toward a specific entity and take insightful actions based on their sentiment.

## CHAPTER - 2

### Literature Survey

#### 2.1 Existing System

Balamurali et al. (2011) presents an innovative idea to introduce sense based sentiment analysis. This implies shifting from lexeme feature space to semantic space i.e. from simple words to their synsets. The works in Sentiment Analysis, for so long, concentrated on lexeme feature space or identifying relations between words using parsing. The need for integrating sense to Sentiment Analysis was the need of the hour due to the following scenarios, as identified by the authors:

- A word may have some sentiment-bearing and some non-sentiment-bearing senses
- There may be different senses of a word that bear sentiment of opposite polarity
- The same sense can be manifested by different words (appearing in the same synset)

Using sense as features helps to exploit the idea of sense/concepts and the hierarchical structure of the WordNet. The following feature representations were used by the authors and their performance were compared to that of lexeme based features:

- A group of word senses that have been manually annotated (M)
- A group of word senses that have been annotated by an automatic WSD (I)
- A group of manually annotated word senses and words (both separately as features) (Sense + Words(M))
- A group of automatically annotated word senses and words (both separately as fea-



tures) (Sense + Words(I))

Sense + Words(M) and Sense + Words(I) were used to overcome non-coverage of Word-Net for some noun synsets. The authors used synset-replacement strategies to deal with non-coverage, in case a synset in test document is not found in the training documents. In that case the target unknown synset is replaced with its closest counterpart among the WordNet synsets by using some metric.

Support Vector Machines were used for classification of the feature vectors and IWSN was used for automatic WSD. Extensive experiments were done to compare the performance of the 4 feature representations with lexeme representation. Best performance, in terms of accuracy, was obtained by using sense based SA with manual annotation (with an accuracy of 90.2 percent and an increase of 5.3 percent over the baseline accuracy) followed by Sense(M), Sense + Words(I), Sense(I) and lexeme feature representation. LESK was found to perform the best among the 3 metrics used in replacement strategies.

One of the reasons for improvements was attributed to feature abstraction and dimensionality reduction leading to noise reduction. The work achieved its target of bringing a new dimension to Sentiment Analysis by introducing sense based Sentiment Analysis.

## CHAPTER - 3

### Requirement Analysis

#### SDLC model for Project (Waterfall Model)

We can choose waterfall model due to nature of this model, it is hard to get back to the previous phase once completed. Although this is can be very rigid in some software projects which need some flexibility while this model can be essential or the most suitable model for other software projects contexts.

The usage of the waterfall model can fall under the projects which do not focus on changing the requirements, for example:

1. Projects initiated from a request for proposal (RFP), the customer has a very clear documented requirements
2. Mission Critical projects, for example, in a Space shuttle
3. Embedded systems.

You can review the selection criteria for the SDLC models from this article, while this is the list of things to consider when using the waterfall:

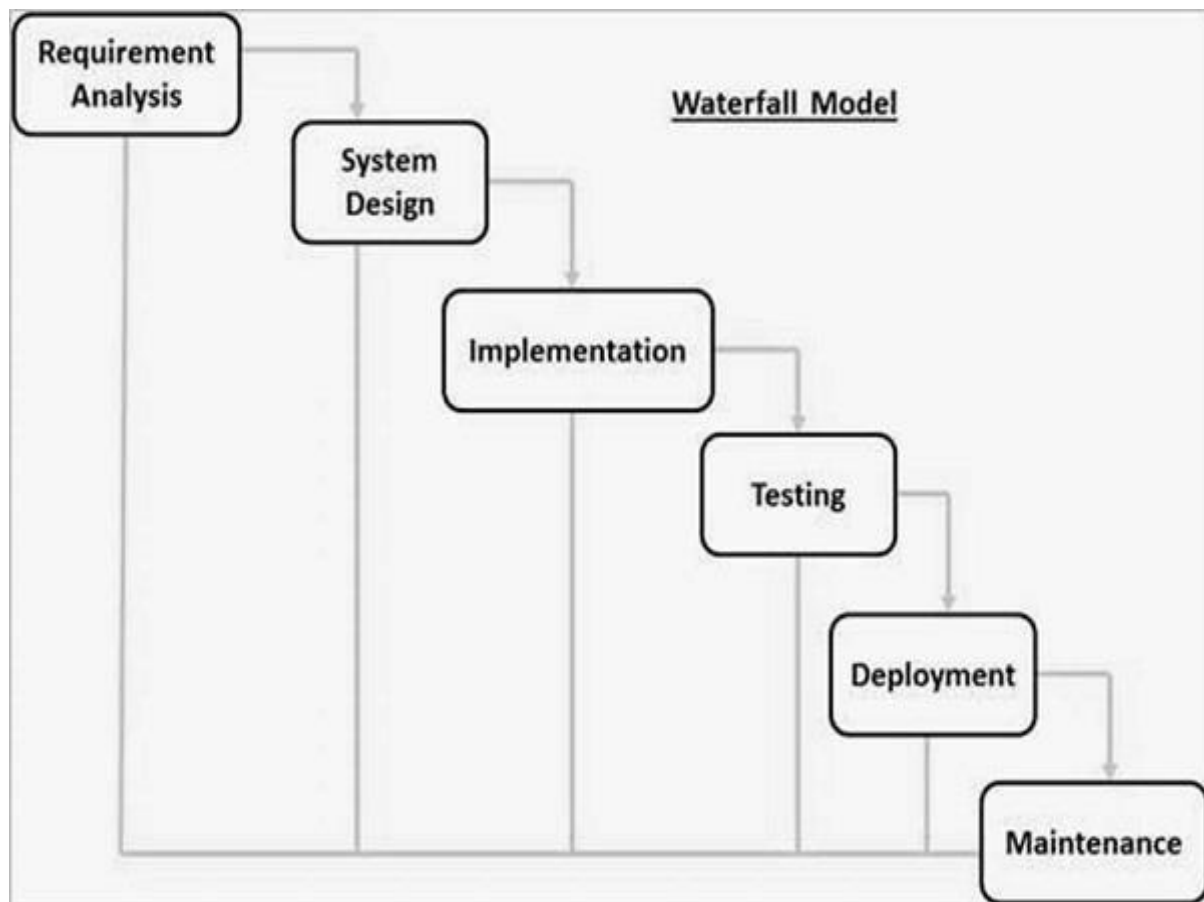
- The requirements are clear and frozen.
- Technology is understood and used by the team in different projects.
- The project cannot be delivered in an iterative manner.
- Documentation is essential.
- Professional Project management skills.
- The project cost is defined.



## Waterfall Model

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



Fig, 3.1 Waterfall Model

### 3.1 Phases Of SDLC Model

- **Requirement Gathering and analysis** –All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** –The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** –With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

## 3.2 Hardware And Software Requirements

### ➤ Hardware Requirements

- I3 or higher
- 256 MB RAM
- 2 Gb hard free drive space

### ➤ Software Requirements

- Python
- Install Textblob
- Operating System: Windows /Linux/ Mac

## CHAPTER - 4

### Project Scheduling

Project management tools are aids to assist an individual or team to effectively organize work and manage projects and tasks. The term usually refers to project management software you can purchase online or even use for free.

There are tools available, which aid for effective project management. A few are described –

#### 4.1 Gantt Chart:

Gantt charts represents project schedule with respect to time periods. It is a horizontal bar chart with bars representing activities and time scheduled for the project activities.

Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13
Study and Analysis													
Data Collection													
Implementation													
Testing													
Documentation													
Review													

Fig.4.1 Gantt Chart

## 4.2 Critical Path Method (CPM):

The critical path method (CPM) is a step-by-step project management technique for process planning that defines critical and non-critical tasks with the goal of preventing time-frame problems and process bottlenecks. The CPM is ideally suited to projects consisting of numerous activities that interact in a complex manner.

In applying the CPM, there are several steps that can be summarized as follows:

- Define the required tasks and put them down in an ordered (sequenced) list.
- Create a flowchart or other diagram showing each task in relation to the others.
- Identify the critical and non-critical relationships (paths) among tasks.
- Determine the expected completion or execution time for each task.
- Locate or devise alternatives (backups) for the most critical paths.

Task	Description	Length	Predecessor
A	Study and Analysis of project	5 week	-
B	Data Collection	1 week	-A
C	Implementation	5 week	-A,B
D	Testing	2 week	-C
E	Documentation	1 week	-D
F	Review	2 week	-D,E

Table 4.1

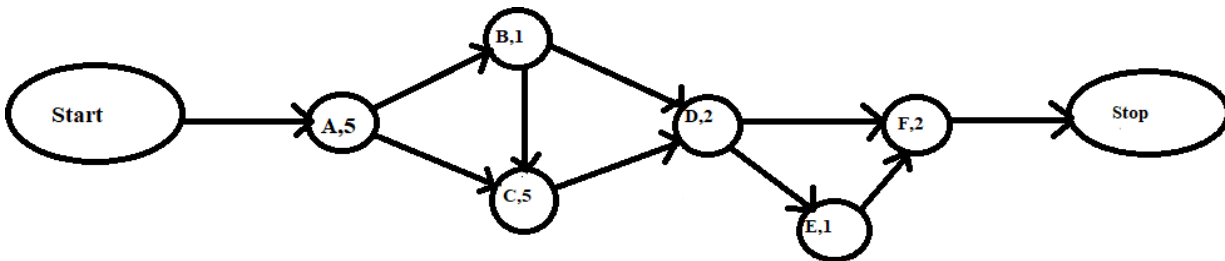


Fig.4.2. Critical Path

Four Path for Start to End:

- 1) A-B-C-D-F =  $5+1+5+2+2 = 15$
- 2) A-B-C-D-E-F =  $5+1+5+2+1+2=16$
- 3) A-C-D-F =  $5+1+2+2=10$
- 4) A-C-D-E-F =  $5+1+2+1+2=11$

Critical Path is: A-B-C-D-E-F = 16 weeks

### 4.3 Earned Value Analysis

Earned Value Analysis (EVA) is an industry standard method of measuring a project's progress at any given point in time, forecasting its completion date and final cost, and analyzing variances in the schedule and budget as the project proceeds.

Earned Value Management measures progress against a baseline. It involves calculating three key values for each activity in the WBS:

1. The Planned Value (PV), (formerly known as the *budgeted cost of work scheduled* or *BCWS*)—that portion of the approved cost estimate planned to be spent on the given activity during a given period.
2. The Actual Cost (AC), (formerly known as the *actual cost of work*

*performed* or *ACWP*)—the total of the costs incurred in accomplishing work on the activity in a given period. This Actual Cost must correspond to whatever was budgeted for the Planned Value and the Earned Value (e.g. all labor, material, equipment, and indirect costs).

3. The Earned Value (EV), (formerly known as the *budget cost of work performed* or *BCWP*)—the value of the work actually completed.

These three values are combined to determine *at that point in time* whether or not work is being accomplished as planned. The most commonly used measures are the cost variance:

Cost Variance (CV) =  
EV - AC and the schedule  
variance:

$$\text{Schedule Variance (SV)} = \text{EV} - \text{PV}$$

These two values can be converted to efficiency indicators to reflect the cost and schedule performance of the project. The most commonly used cost-efficiency indicator is the cost performance index (CPI). It is calculated thus:

$$\text{CPI} = \text{EV} / \text{AC}$$

The sum of all individual EV budgets divided by the sum of all individual AC's is known as the cumulative CPI, and is generally used to forecast the cost to complete a project.

The schedule performance index (SPI), calculated thus:

$$\text{SPI} = \text{EV} / \text{PV}$$

is often used with the CPI to forecast overall project completion estimates.

A negative schedule variance (SV) calculated at a given point in time means the project is behind schedule, while a negative cost variance (CV) means the project is over budget.



## CHAPTER - 5

### Requirement Modeling

#### 5.1 Object Oriented Requirement Modeling

##### 5.1.1 Use Case Diagram

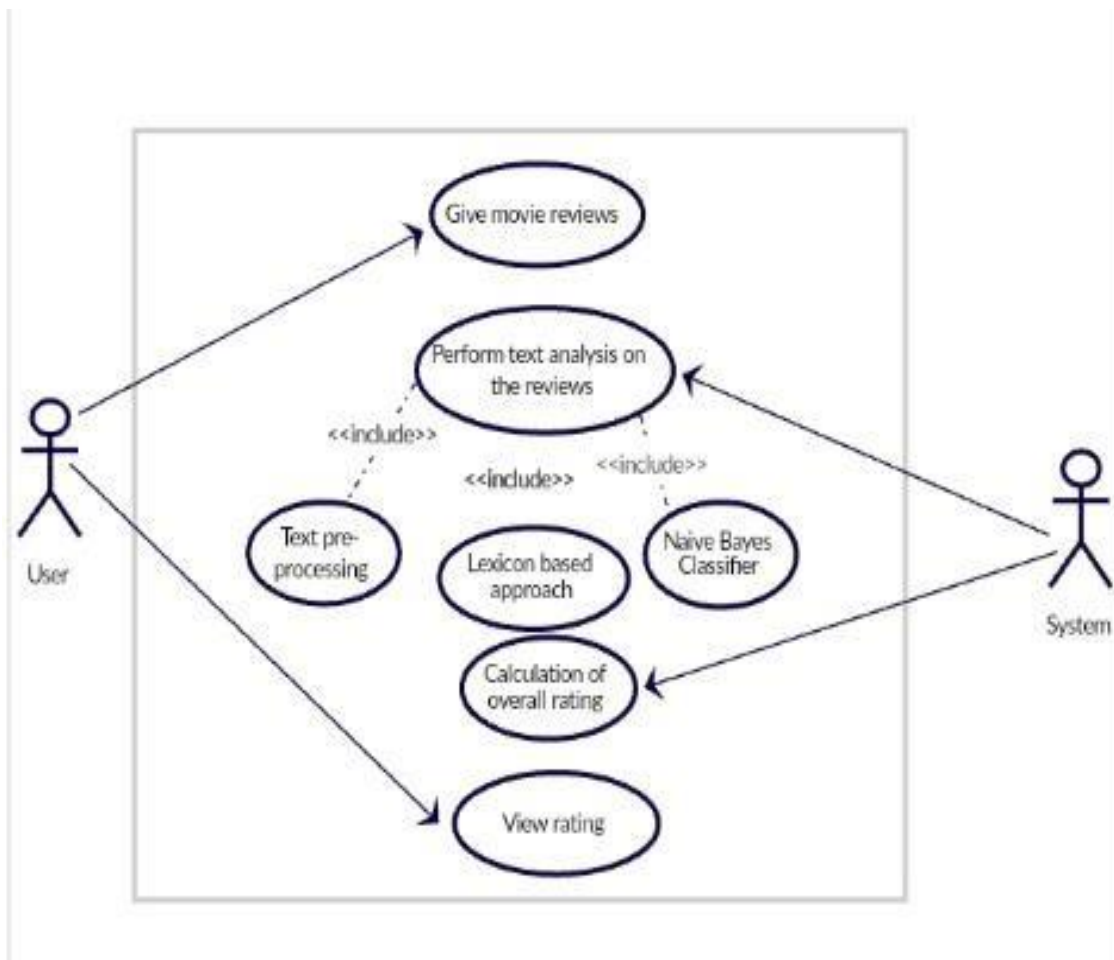


Fig.5.1 Use Case Diagram

### 5.1.2 Class Diagram

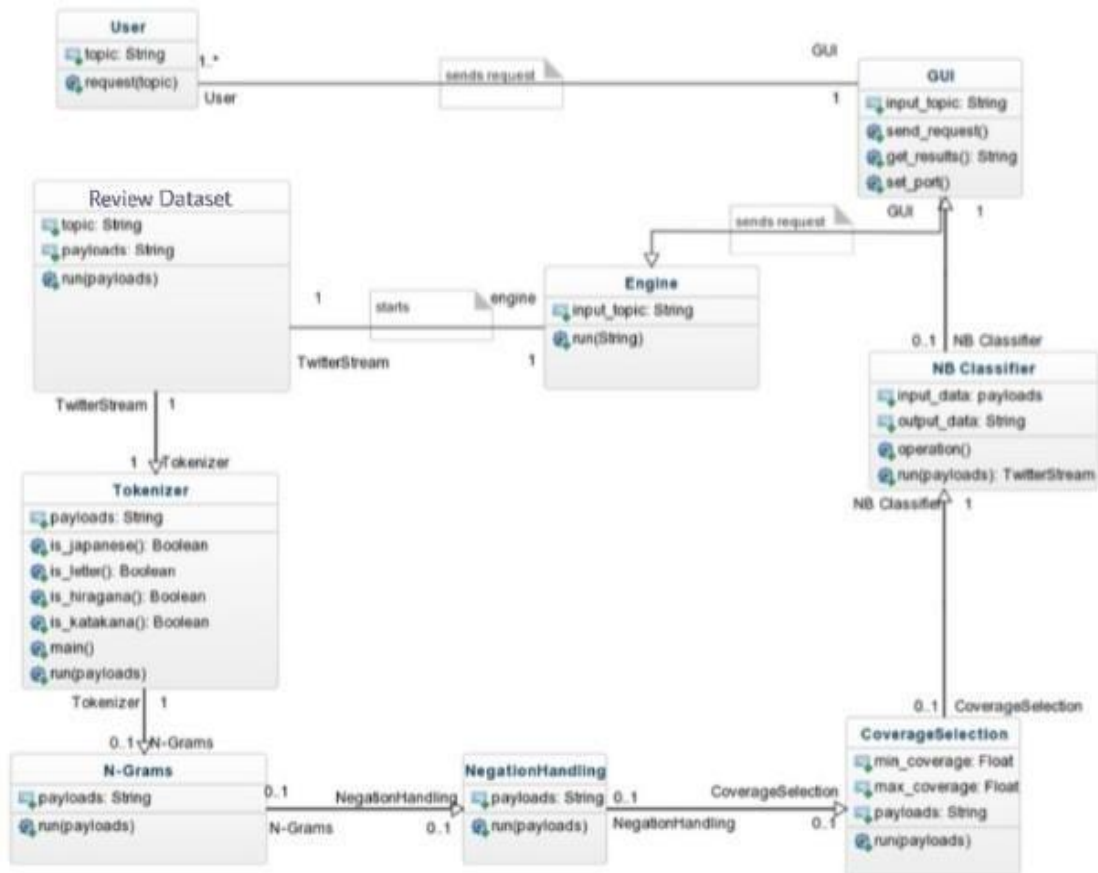
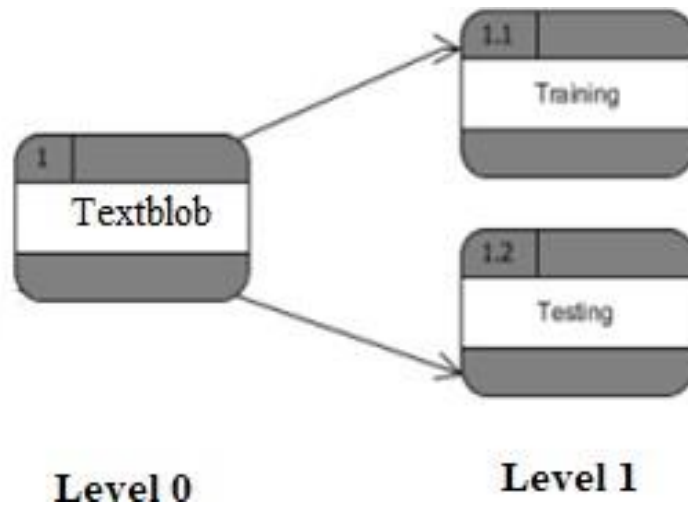


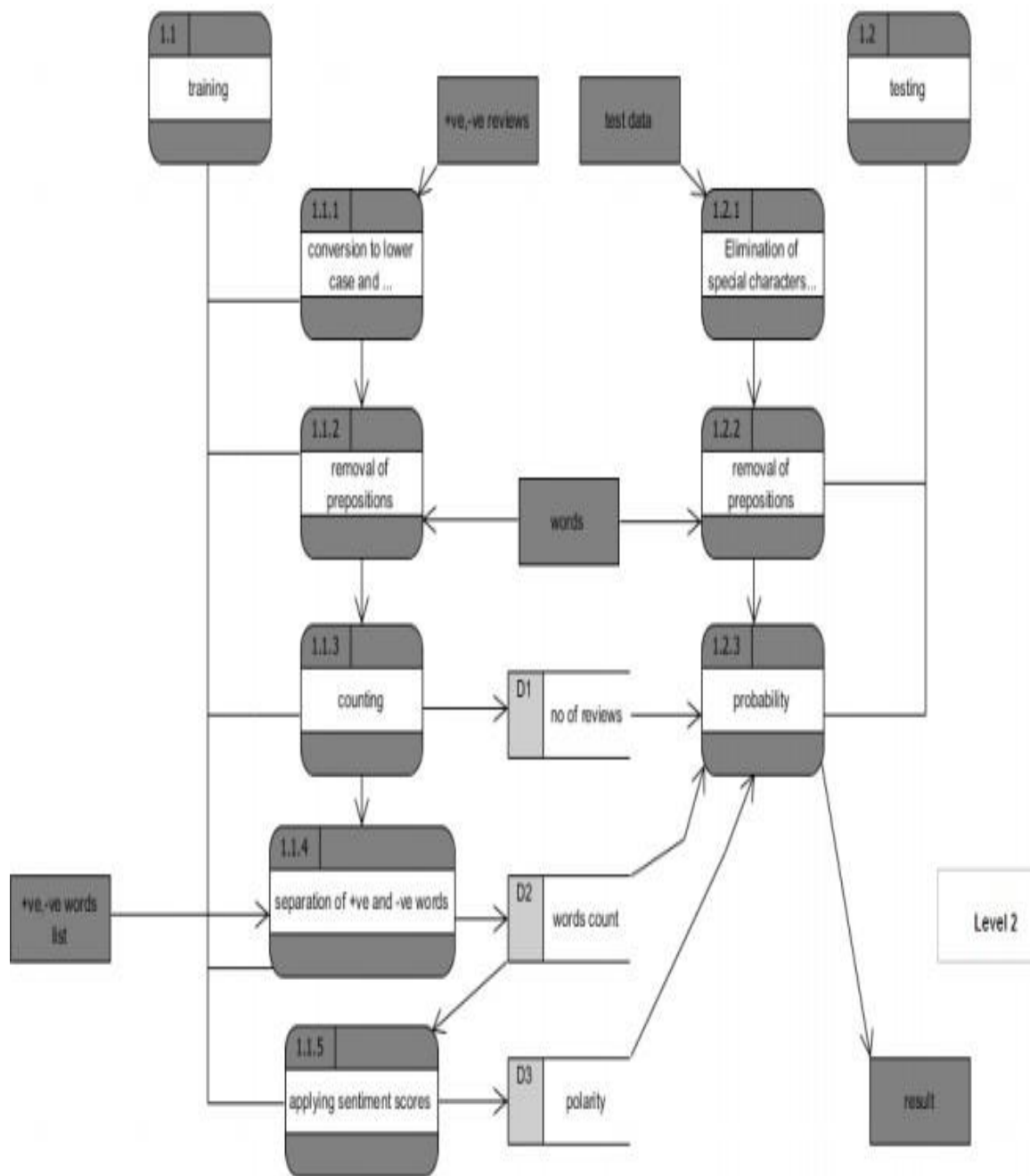
Fig.5.2 Class Diagram

## 5.2 Structured Requirement Modeling

### 3.2.1 Data Flow Diagram



Level 0 and Level 1 Data Flow Diagram



Level 2 Data Flow Diagram for the process 1(Naïve Bayes)

## CHAPTER - 6

### Methodology

#### 6.1 Project Modules

##### Setting up the System

Installation of TextBlob in your system in a simple task, all you need to do is open anaconda prompt ( or terminal if using Mac OS or Ubuntu) and enter the following commands:

```
pip install -U textblob
```

This will install TextBlob. For the uninitiated – practical work in Natural Language Processing typically uses large bodies of linguistic data, or **corpora**. To download the necessary corpora, you can run the following command

```
python -m textblob.download_corpora
```

##### NLP tasks using TextBlob

###### 6.1.1 Tokenization

Tokenization refers to dividing text or a sentence into a sequence of tokens, which roughly correspond to “words”. This is one of the basic tasks of NLP. To do this using TextBlob, follow the two steps:

1. Create a **textblob** object and pass a string with it.
2. Call **functions** of textblob in order to do a specific task.

So, let's quickly create a textblob object to play with.

```
from textblob import TextBlob
```

```
blob = TextBlob("Analytics Vidhya is a great platform to learn data science. \n It helps community through blogs, hackathons, discussions,etc.")
```

### 6.1.2 Noun Phrase Extraction

Since we extracted the words in the previous section, instead of that we can just extract out the noun phrases from the textblob. Noun Phrase extraction is particularly important when you want to analyze the “who” in a sentence. Lets see an example below.

```
blob = TextBlob("Analytics Vidhya is a great platform to learn data science.")
for np in blob.noun_phrases:
    print(np)
>> analytics vidhya
great platform
data science
```

As we can see that the results aren't perfectly correct, but we should be aware that we are working with machines.

### 6.1.3 Part-of-speech Tagging

Part-of-speech tagging or grammatical tagging is a method to mark words present in a text on the basis of its definition and context. In simple words, it tells whether a word is a noun, or an adjective, or a verb, etc. This is just a complete version of noun phrase extraction, where we want to find all the the parts of speech in a sentence.

Let's check the tags of our textblob.

for words, tag in blob.tags:

```
print (words, tag)
```

```
>> Analytics NNS
```

```
Vidhya NNP
```

```
is VBZ
```

```
a DT
```

```
great JJ
```

```
platform NN
```

```
to TO
```

```
learn VB
```

```
data NNS
```

```
science NN
```

Here, NN represents a noun, DT represents as a determiner, etc. You can check the full list of tags from [here](#) to know more.

### 6.1.4 Words Inflection and Lemmatization

*Inflection* is a process of *word* formation in which characters are added to the base form of a *word* to express grammatical meanings. Word inflection in TextBlob is very simple, i.e., the words we tokenized from a textblob can be easily changed into singular or plural.

```
blob = TextBlob("Analytics Vidhya is a great platform to learn data science. \n It helps community
through blogs, hackathons, discussions,etc.")
print (blob.sentences[1].words[1])
print (blob.sentences[1].words[1].singularize())
```

```
>> helps
help
```

TextBlob library also offers an in-build object known as *Word*. We just need to create a word object and then apply a function directly to it as shown below.

```
from textblob import Word
w = Word('Platform')
w.pluralize()
>>'Platforms'
```

We can also use the tags to inflect a particular type of words as shown below.

```
## using tags
for word,pos in blob.tags:
    if pos == 'NN':
        print (word.pluralize())
>> platforms
```



sciences

Words can be lemmatized using the *lemmatize* function.

```
## lemmatization
w = Word('running')
w.lemmatize("v") ## v here represents verb
>> 'run'
```

### 6.1.5 N-grams

A combination of multiple words together are called N-Grams. N grams ( $N > 1$ ) are generally more informative as compared to words, and can be used as features for language modelling. N-grams can be easily accessed in TextBlob using the *ngrams* function, which returns a tuple of n successive words.

```
for ngram in blob.ngrams(2):
    print (ngram)
>> ['Analytics', 'Vidhya']
['Vidhya', 'is']
['is', 'a']
['a', 'great']
['great', 'platform']
['platform', 'to']
['to', 'learn']
['learn', 'data']
['data', 'science']
```

### 6.1.6 Sentiment Analysis

Sentiment analysis is basically the process of determining the attitude or the emotion of the writer, i.e., whether it is positive or negative or neutral.

The *sentiment* function of textblob returns two properties, **polarity**, and **subjectivity**.

Polarity is float which lies in the range of  $[-1,1]$  where 1 means positive statement and -1 means a negative statement. Subjective sentences generally refer to personal opinion, emotion or judgment whereas objective refers to factual information. Subjectivity is also a float which lies in the range of  $[0,1]$ .

Let's check the sentiment of our blob.

```
print(blob)
blob.sentiment
>> Analytics Vidhya is a great platform to learn data science.
Sentiment(polarity=0.8, subjectivity=0.75)
```

We can see that polarity is **0.8**, which means that the statement is positive and **0.75** subjectivity refers that mostly it is a public opinion and not a factual information.

## 6.2 Library used

Natural Language Processing approach uses SentiWordNet lexicon. Which consists of positive, negative score for each of the term occurring in WordNet. The implementation done by extracting the adjectives out of the sentence and then searching it in the SentiWordNet to find out its positive, negative score. In this way the total net score of the sentence is calculated and whichever is greater (either positive or negative) becomes the review for the sentence.



Following figure shows the basic implementation architecture of Sentiment Analysis using Natural Language Processing Approach.

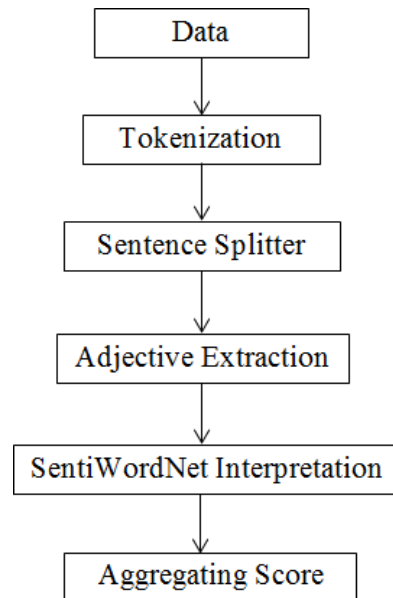


Figure 6.1: Implementation Architecture using NLP Approach

### About TextBlob?

TextBlob is a python library and offers a simple API to access its methods and perform basic NLP tasks.

A good thing about TextBlob is that they are just like python strings. So, you can transform and play with it same like we did in python. Below, I have shown you below some basic tasks. Don't worry about the syntax, it is just to give you an intuition about how much-related TextBlob is to Python strings.

```
string1 = TextBlob("Analytics")  
string1[1:5]    ### extracting 1 to 5 letters  
TextBlob("naly")  
  
string1.upper() ## to upper case the entire text  
TextBlob("ANALYTICS")  
  
string2 = TextBlob("Vidhya")  
  
## concat two sentences similar as python  
string1 + " " + string2  
TextBlob("Analytics Vidhya")
```

So, to perform these things on your own let's quickly install and start coding.

## CHAPTER - 7

### Implementation

#### 7.1 Library Used

##### TextBlob Sentiment: Calculating Polarity and Subjectivity

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
```

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

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.

But where do these numbers come from?

Let's find out by going to the source. (This will refer to sloria/TextBlob on GitHub at commit eb08c12.)

After digging a bit, you can find that the main default sentiment calculation is defined in `_text.py`, which gives credit to the pattern library. (I'm not sure how much is original and how much is from pattern.)

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_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”, 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. 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)
```

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! 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 you can see that subjectivity is also modified by “very” to become  $0.75 \cdot 1.3 = 0.975$

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)
```

How's that?

$\text{polarity} = -0.5 \cdot 1.3 \cdot 0.8 \approx -0.31$

$\text{subjectivity} = 1.3 \cdot 0.75 \approx 0.58$

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.

## CHAPTER - 8

### Results And Discussions

#### 8.1 System Program:

```
from textblob import TextBlob
```

```
pos_count = 0
```

```
pos_correct = 0
```

```
with open("positive.txt","r") as f:
```

```
    for line in f.read().split("\n"):
```

```
        analysis = TextBlob(line)
```

```
        if analysis.sentiment.polarity > 0:
```

```
            pos_correct += 1
```

```
        pos_count +=1
```

```
neg_count = 0
```

```
neg_correct = 0
```

```
with open("negative.txt","r") as f:
```

```
    for line in f.read().split("\n"):
```

```
        analysis = TextBlob(line)
```

```
        if analysis.sentiment.polarity <= 0:
```

```
            neg_correct += 1
```

```
        neg_count +=1
```

```
print("accuracy by fairly positive, but also highly subjective")
```

```
print("Positive accuracy = {}% via {  
} samples".format(pos_correct/pos_count*100.0, pos_count))  
print("Negative accuracy = {}% via {  
} samples".format(neg_correct/neg_count*100.0, neg_count))
```

```
pos_count = 0  
pos_correct = 0
```

```
with open("positive.txt","r") as f:  
    for line in f.read().split("\n"):  
        analysis = TextBlob(line)  
  
        if analysis.sentiment.subjectivity < 0.3:  
            if analysis.sentiment.polarity > 0:  
                pos_correct += 1  
                pos_count +=1
```

```
neg_count = 0  
neg_correct = 0
```

```
with open("negative.txt","r") as f:  
    for line in f.read().split("\n"):  
        analysis = TextBlob(line)  
        if analysis.sentiment.subjectivity < 0.3:  
            if analysis.sentiment.polarity <= 0:  
                neg_correct += 1  
                neg_count +=1
```

```
print("accuracy by increasing more objective")
print("Positive accuracy = { }% via {
} samples".format(pos_correct/pos_count*100.0, pos_count))
print("Negative accuracy = { }% via {
} samples".format(neg_correct/neg_count*100.0, neg_count))
```

```
pos_count = 0
pos_correct = 0
```

```
with open("positive.txt", "r") as f:
    for line in f.read().split("\n"):
        analysis = TextBlob(line)

        if analysis.sentiment.subjectivity > 0.9:
            if analysis.sentiment.polarity > 0:
                pos_correct += 1
                pos_count += 1
```

```
neg_count = 0
neg_correct = 0
```

```
with open("negative.txt", "r") as f:
    for line in f.read().split("\n"):
        analysis = TextBlob(line)

        if analysis.sentiment.subjectivity > 0.9:
            if analysis.sentiment.polarity <= 0:
```

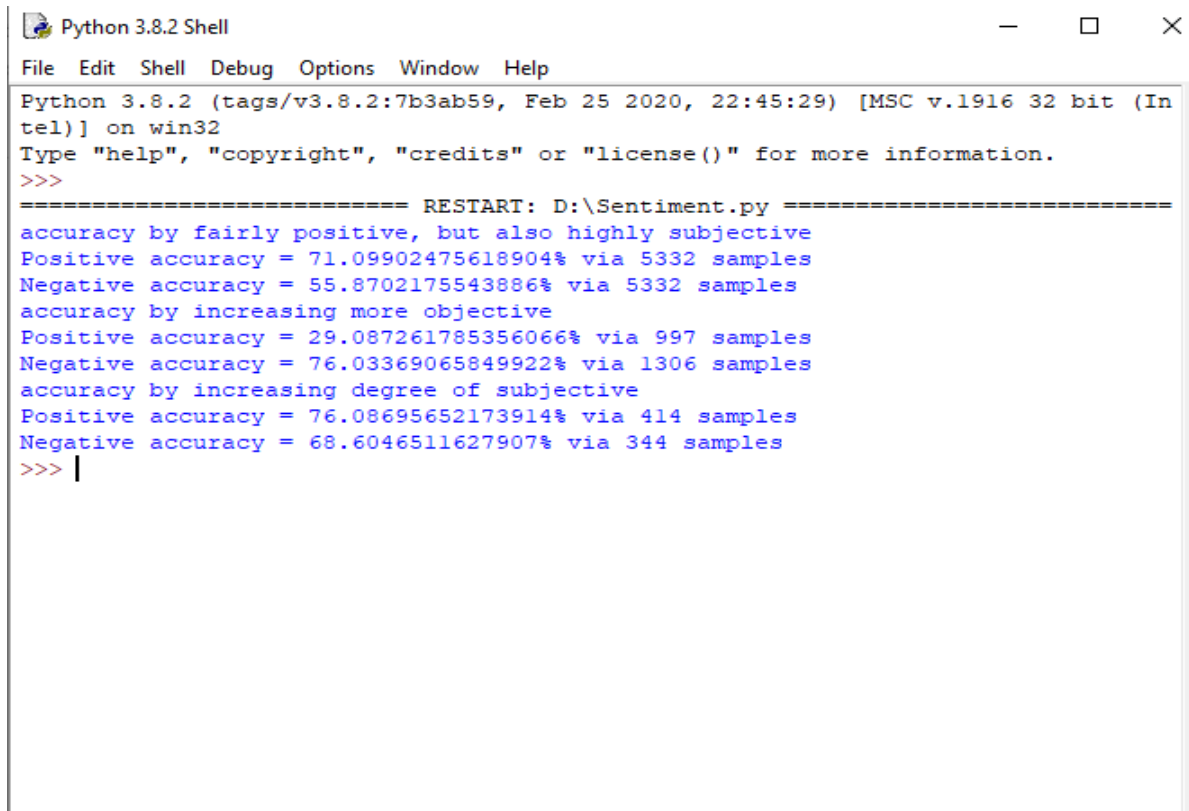
```

        neg_correct += 1
    neg_count += 1

print("accuracy by increasing degree of subjective")
print("Positive accuracy = {}% via {
} samples".format(pos_correct/pos_count*100.0, pos_count))
print("Negative accuracy = {}% via {
} samples".format(neg_correct/neg_count*100.0, neg_count))

```

## 8.2 Working Of The System (Screen Snapshots and details of the same)



```

Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Sentiment.py =====
accuracy by fairly positive, but also highly subjective
Positive accuracy = 71.09902475618904% via 5332 samples
Negative accuracy = 55.8702175543886% via 5332 samples
accuracy by increasing more objective
Positive accuracy = 29.087261785356066% via 997 samples
Negative accuracy = 76.03369065849922% via 1306 samples
accuracy by increasing degree of subjective
Positive accuracy = 76.08695652173914% via 414 samples
Negative accuracy = 68.6046511627907% via 344 samples
>>> |

```



## CHAPTER - 8

### Conclusions

Sentiment analysis, as an interdisciplinary field that crosses natural language processing, artificial intelligence, and text mining. We have seen that Sentiment Analysis can be used for analyzing opinions in blogs, newspaper, articles, Product reviews, Social Mediawebsites, Movie-review websites where a third person narrates his/her views. We also studied Natural Language Processing and Machine Learning approaches for Sentiment Analysis. We have seen that is easy to implement Sentiment Analysis via *SentiWordNet* approach than via *Classifier* approach. We have seen that sentiment analysis has many applications and it is important field to study. Sentiment analysis has *Strong commercial interest* because Companies want to know how their products are being perceived and also Prospective consumers want to know what existing users think



## References

- [1] P. W. V.K. Singh R. Piryani A. Uddin, "Sentiment analysis of movie reviews and blog posts," *IEEE International Advance Computing Conference (IACC)*, vol. 3, 2013.
- [2] A. A. G. Mostafa Karamibekr, "Sentiment analysis of social issues," *International Conference on Social Informatics*, 2012.
- [3] M. R. Alaa Hamouda, "Reviews classification using sentiwordnet lexicon," *The Online Journal on Computer Science and Information Technology (OJCSIT)*, vol. 2, August 2011.