DWIT COLLEGE

DEERWALK INSTITUTE OF TECHNOLOGY

Tribhuvan University

Institute of Science and Technology

SENTIMENT ANALYSIS OF MOVIE REVIEWS

A PROJECT REPORT

Submitted to

Department of Computer Science and Information Technology

DWIT College

In partial fulfillment of the requirements for the Bachelor's Degree in Computer Science and Information Technology

Submitted by

Anish Thakuri

Bidish Acharya

August, 2016

DWIT College

DEERWALK INSTITUTE OF TECHNOLOGY

Tribhuvan University

SUPERVISOR'S RECOMENDATION

I hereby recommend that this project prepared under my supervision by ANISH THAKURI and BIDISH ACHARYA entitled **"SENTIMENT ANALYSIS OF MOVIE REVIEWS"** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology be processed for the evaluation.

……

Sarbin Sayami

Assistant Professor

IOST, Tribhuvan University

DWIT College

DEERWALK INSTITUTE OF TECHNOLOGY

Tribhuvan University

LETTER OF APPROVAL

This is to certify that this project prepared by ANISH THAKURI and BIDISH ACHARYA entitled **"SENTIMENT ANALYSIS OF MOVIE REVIEWS"** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology has been well studied. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

…… Sarbin Sayami [Supervisor] Assistant Professor IOST, Tribhuvan University	…… Hitesh Karki Chief Academic Officer DWIT College
…….. Jagadish Bhatta [External Examiner] IOST, Tribhuvan University	…….. Rituraj Lamsal [Internal Examiner] Lecturer DWIT College

ACKNOWLEDGEMENT

This project would not have been possible without the joint efforts of many individuals. We would like to express our sincere thanks to all who were involved in this project. We owe special thanks to a number of people, whose time and expertise has brought a lot of value in this project and it would have been very difficult for us to complete our project without them.

We are highly indebted to our project supervisor **Mr. Sarbin Sayami,** Asst. Professor, Tribhuvan University, for his valuable suggestions and constant supervision to help us take our project forward. His help in providing required information and continuous guidance was the huge support for our project.

We would like to extend our special thanks to **Mr. Bikash Balami** Sir for providing us with technical support and insights on different sentiment analysis projects, which was significant for our project to foster.

We would also like to thank **Mr. Hitesh Karki,** CAO, DWIT College, for his constant feedback on our project and guiding us throughout the project. We would like to thank our friend **Mr. Sachin Aryal** for willingly helping us out in developing the project.

At the end, we would like to thank all our friends who have directly or indirectly helped us in our project.

Anish Thakuri

TU Exam Roll no: 1790/069

Bidish Acharya

TU Exam Roll no: 1797/069

STUDENT'S DECLARATION

We hereby declare that we are the only authors of this work and that no sources other than the listed here have been used in this work.

.....
Anish Thakuri

TU Exam Roll no: 1790/069

.....

Bidish Acharya

TU Exam Roll no: 1797/069

Date: .....

ABSTRACT

Sentiment Analysis is a Natural Language Processing task to identify opinions expressed in a source material. There has been a lot of research in the field of Sentiment Analysis. Nowadays, as a lot of people voice their opinion on different social media sites, opinion mining has been very talked about topic, as it helps to analyze those opinion and generate some valuable information from it.

In this project, sentiment analysis of movie reviews has been performed by comparing the polarity of input text with the polarity of the words that are stored in the system, using the baseline model. This project has used movie reviews as the dataset. The system generates the result based on the text provided and displays it on the web-page deciding if the movie review is either positive, negative, or neutral.

Keywords**:** Sentiment Analysis, Natural Language Processing, baseline model, opinion mining

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SA	Sentiment Analysis
NLP	Natural Language Processing
HTML	Hyper Text Markup Language
JSP	Java Servlet Page
IMDB	The Internet Movie Database
CSS	Cascading Style Sheet
UML	Unified Modelling Language

CHAPTER 1: INTRODUCTION

1.1 Background

Sentiment Analysis is a Natural Language Processing task that is used to identify opinion expressed by the public in a certain source material. It is also described as the process of getting insights from texts from different sources such as Facebook posts, tweets, blogs, online news portals, etc. Those insights can be analyzed and we can classify the sentiments as either positive or negative.

In this project, sentiment analysis will be implemented on the movie reviews to determine if the review is either negative or positive. For this, content from different news sites, blog posts, or from social media sources will not be used. This project will use a movie review data set from IMDB and test our application using that data set.

1.2 Problem Statement

Sentiment analysis is a very trending topic currently and it helps make analyzing texts easier. Sentiment Analysis has become extremely helpful in collecting and analyzing a lot of publics' opinion, which leads us to a conclusion, based on certain problem statement.

When a new movie hits the theater, people go through the reviews that are posted online and make decisions based on that. As online blogs are getting more and more attention day by day and as many people have started exposing their opinion publicly on the internet, those opinion can be helpful for everyone to make decisions. For example, if the user wants to watch the movie "Kabaddi", the user can go through the online reviews and see how other people perceive the movie, and make the decision if they really want to watch it.

However, this is a manual process and can be hectic. This application proves to be helpful in this scenario, as it processes the text provided by the user and will analyze if the reviews are actually positive or negative.

This will help people to know the review of the movie without even having to read all the reviews.

1.3 Objectives

A) To determine the sentiment of the text provided by the user.
B) To classify the sentiment as positive or negative.

1.4 Scope

This project can be used by people to see if the movie reviews are either positive or negative. This project will become helpful to the public who wants to watch a certain movie and needs to know if the review is good or not, without having to manually go through different reviews.

1.5 Limitation

A) All the words in the dictionary cannot be stored and provided with sentiment polarity.

1.6 Outline

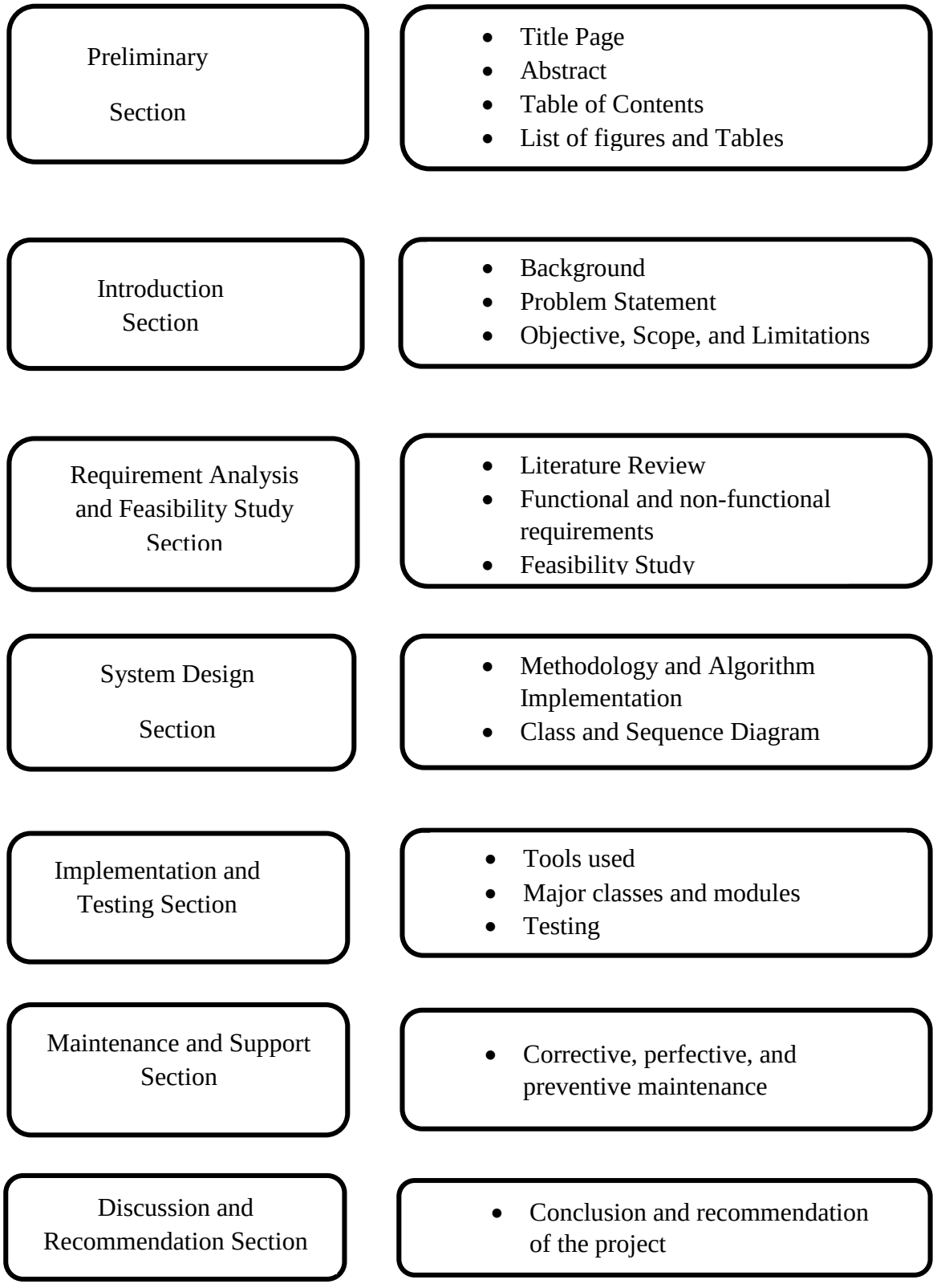


Figure 1 – Project block diagram

CHAPTER 2: REQUIREMENT ANALYSIS AND FEASIBILITY

2.1 Literature Review

Sentiment Analysis is not a new topic in itself, a lot of research has already been done in the field of Sentiment Analysis. As online blogs are getting more attention day by day and as many people have started exposing their opinion publicly on the internet, those opinions can be helpful for everyone to make decisions. For example, if people want to watch the movie "Kabaddi", they can go through the online reviews and see how other people perceive the movie, and make the decision if they really want to watch it.

In 'Fast and accurate sentiment classification using an enhanced Naïve Bayes model', Vivek Narayanan, Ishan Arora, and Arjun Bhatia talks about how the accuracy of a Naïve Bayes Classifier for sentiment analysis can be enhanced by implementing a combination of methods like effective negation handling, word n-grams and feature selection. They have implied that a highly accurate and fast sentiment classifier can be used by using a simple Naïve Bayes model with a combination of other methods. They have found an accuracy of 88.80% on the popular IMDB movie reviews dataset. The Table 1 signifies how the accuracy of classifier was improved by implementing those methods [1].

Table 1 - Comparison of accuracy of Naïve Bayes classifier by implementing various methods

Feature Added	Accuracy on test set
Original Naïve Bayes algorithm with Laplacian Smoothing	73.77%
Handling negations	82.80%
Bernoulli Naïve Bayes	83.66%
Bigrams and trigrams	85.20%
Feature Selection	88.80%

In 'Sentiment Analysis on Movie Reviews using Recursive and Recurrent Neural Network Architectures', Aditya Timmaraju and Vikesh Khanna talks about various methods that can be implemented to perform sentiment analysis. Sentiment Classification is traditionally solved using linear classification methods, such as Support Vector Machines (SVM) and logistic regression. They tried to propose an architecture that performs well with word vector representations. They proposed a new architecture that encapsulates previous work applied to sentence-level sentiment classification, using Recursive Neural Networks and use it in unison with a Recurrent Neural Network [11].

They have implemented 6 different approaches and compared their accuracies. The different approaches that they have considered are:

1. Using Semantic Word Vectors and Recurrent Architecture
2. Recursive Neural Network with mean likelihood
3. Recursive Neural Network with Affine NN
4. Averaged Semantic Word Vectors
5. Semantic Word Vectors and Bag-of-Words Features
6. Recursive-Recurrent Neural Network Architecture

Table 2 - Comparison of various approaches used in sentiment analysis

Approach	**Test Accuracy**
Method a, Mean Word – RecNN	82.35%
Method b, Mean Prob – RNN	81.8%
Method c, RNN – Affine	81.42%
Method d, SVM – RBF	76.69%
Method d, SVM – Linear	83.28%
Method e, SVM – Linear	**86.496%**
Method e, 2-Layer NN	83.94%
Method f, RecNN – RNN	83.88%

They concluded that a simple model with handcrafted features (like Bag of words with TF-IDF, bigrams, etc.) performs better than other methods. So, using features like TF-IDF vectors, bigrams, etc. will enhance the accuracy of the sentiment classifier [11].

2.2 Requirement Analysis

A requirement analysis was carried out while performing the analysis. The outcomes of the analysis are mentioned below:

2.2.1 Functional requirements

This application is mainly focused on implementing an algorithm for sentiment analysis, in which the user provides the text to the application and gets the result if the review is either positive or negative. Hence, the functional requirements of this product can be defined by a user.

User

In this application, user is the one who uploads the reviews of the movies as a text. The text is then processed based on our algorithm, and the result is displayed back to the user.

Functions of the user

A) Users can enter the URL of the system and access the application.
B) After the URL is entered, web server sends the landing page of the application to the browser.
C) User then writes the text on the text area of the web-page and submits.
    i. The text is read by the server and performs tokenization on the provided text. Tokenization breaks down the stream of texts into words.
    ii. The tokens are then sent to stop-words removal class, which compares the tokens with the file consisting of all the stop-words. If the tokens match, they are removed as they provide no significance in sentiments of the writer.
    iii. Then, Porter Stemming of the texts is performed. Here, multiple words with the same meaning are removed. For ex: we keep only ‘fascinate’ for fascinating, fascinated, fascinates, etc. Also, the plurals and suffixes of the words are removed.
    iv. After stemming the date, the polarity of the words are identified by comparing with a hash table that stores the polarity values of large number of words.
    v. The polarity is calculated, and the sentiment of the text is identified. If the polarity of the text provided by the user is positive, then the review is positive, and vice versa.

D) The result is displayed on the browser denoting if the review of the movie is either positive or negative.

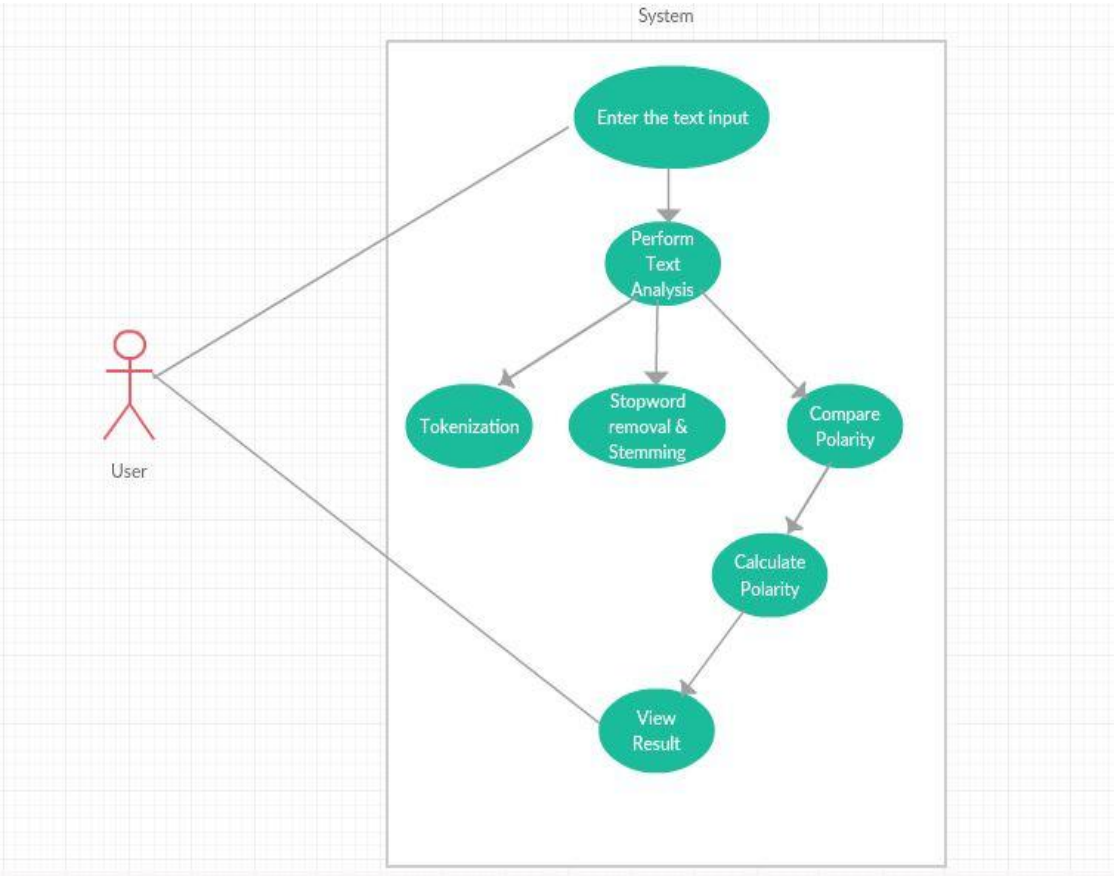


Figure 2 - Use case diagram of the application

2.2.2 Non-functional requirement

Non-functional requirements are also essential for our product, to make sure that the application performs better at all times.

A) Get the data from online resources to test the application.
B) The system will be designed to give high-performance.
C) Specifications and configuration setting of external libraries and API's should be thoroughly studied while integrating with the user application.

2.3 Feasibility Analysis

2.3.1 Operational feasibility

The application to implement sentiment analysis has a simple two-tier architecture (i.e. a client and a server), and it is easy to use.

The application can be accessed from anywhere if there is a working internet connection.

2.3.2 Technical feasibility

Sentiment Analysis of Movie Reviews is a web application that uses Vertex Framework. It uses HTML/CSS, as front end and Java as the back end. It requires a server, client and internet connection to function properly. It supports both Windows and Linux platform for its operation. All of the technologies required by the project are easily available and can be accessed freely. This determines that it is technically feasible.

2.3.3 Schedule feasibility

As shown in figure 2, it was determined that the project was scheduled to complete within the 14 weeks of a semester. Hence, it was identified to be feasible in terms of schedule.

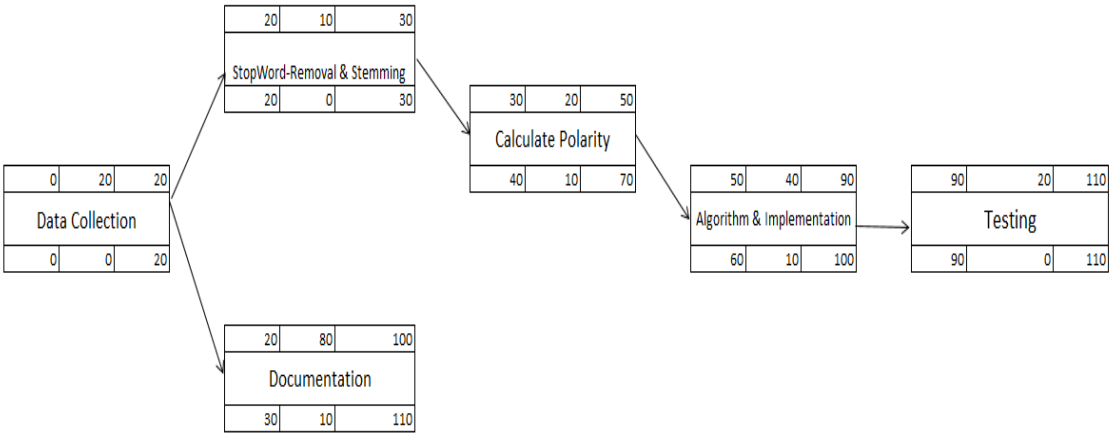


Figure 3 - Activity network diagram of the project

CHAPTER 3: SYSTEM DESIGN

3.1 Methodology

The data was collected from polarity dataset v2.0, which was used in "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts" by Bo Pang Lillian Lee. This dataset consists of 1000 positive and 1000 negative data of different movies [4].

Firstly, tokenization was performed on the input text by using different parsing methods. Then, stop-words removal is performed on the input text using to remove the insignificant words from the text provided. Stop-words do not provide much significance in analyzing the sentiment of the reviews, so those words were removed.

Then, Porter Stemming was performed on the processed texts. Porter Stemming helps to remove redundant use of words that mean the same thing. This means that instead of using fascinating, fascinated, fascinates, etc., only 'fascinate' could be used.

Then, polarity of the words in the text document is checked with the polarity of the words in hash table, where large set of words are stored with their respective polarities. Lastly, the polarity of each word in the text is calculated and the total polarity of the full text is calculated. If the polarity of the text is positive, then the text provided will get a positive review. In this way, the final output of the analysis was generated.

3.1.1 Data collection

The data was collected from polarity dataset v2.0, which was used in "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts" by Bo Pang Lillian Lee. The dataset has a list of 1000 positive and 1000 negative

reviews provided on IMDB. Those data will be used to perform test on the developed application [4].

Sample Input Data

It appears Deadwood and The Avengers had a baby and produced this film, unfortunately all the good traits appear to be lost in this mash up super hero/ Villain tripe!

3.1.2 Tokenization

Firstly, the stream of text are broken up into words, known as tokens. Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing such as parsing or text mining. It helps to break the word from the sentence.

3.1.3 Stop word removal

In computing, stop words are words which are filtered out before or after processing of natural language data [6]. Stop words usually refer to the most common word in the NLP that helps to connect the words together to form meaningful sentence. Example is shown in figure 4.

a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

Figure 4 - Different examples of stop-words

Generally, in performing sentiment analysis, stop words are filtered out before processing the data.

3.1.4 Porter stemming

Porter Stemming is the technique to bring all the related words (verb) to a recognizable verb form. In general, it is the morphological process for removing common morphological endings from words in English [7]. It is mainly used to remove the redundant words that give the same meaning. For example, played, playing, plays, etc. are different forms of word play. So, instead of keeping all of those words, we represented all the words with a single word, i.e. play, by performing Porter Stemming.

Negation handling is one of the factors that contributed significantly to the accuracy of our classifier. A major problem faced during the task of sentiment classification is that of handling negations. Since, we are using each word as feature, the word "good" in the phrase "not good" will be contributing to positive sentiment rather that negative sentiment as the presence of "not" before it is not taken into account. To solve this problem, a simple algorithm for handling negations using state variables and bootstrapping was devised. The idea was to use an alternate representation of negated forms. It transforms word followed by a not or n't into "not_" + word. Whenever the negation state variable is set, the words read are treated as "not_" + word. The state variable is reset when a punctuation mark is encountered or when there is double negation [1].

3.1.5 Polarity of words

After performing Porter Stemming on the data, the polarity of words are identified. The application consists of the value of polarity of large number of words in a Hash Table. A hash table (hash map) is a data structure used to implement an associative array, a structure that can map keys to values. Polarity is constructed based on the words present in the sentence. If the polarity is more inclined to positive value, then the review is a positive sentiment. Similarly, if the polarity is more inclined to negative value, then it is considered as a negative review.

3.1.6 Algorithm

The application consists of a hash table consisting of large amount of words, which stores words and its' polarity value in key, value pair. Words with positive polarity are inclined to positive sentiment, whereas the words with negative polarity are inclined to negative sentiment.

For the classification algorithm, Naive Bayes classifier is used. A Naive Bayes classifier works by figuring out the probability of different attributes of the data being associated with a certain class. This is based on the Bayes' theorem. The theorem states:

P (A|B) = P ((B|A) P (A))/ P (B)

This basically states "the probability of A given that B is true equals the probability of B given that A is true times the probability of A being true, divided by the probability of B being true" [10].

Naive Bayes extends Bayes' theorem to handle this case by assuming that each data point is independent.

The formula looks like this:

P(y|x1,…,xn)=P(y)∏ni=1P(xi|y)P(x1,…,xn)P(y|x1,…,xn)=P(y)∏i=1nP(xi|y)P(x1,…,xn)

This is saying "the probability that classification y is correct given the features x1x1, x2x2, and so on equals the probability of y times the product of each x feature given y, divided by the probability of the x features".

We want to predict whether a review is negative or positive given only the text.

After performing tokenization and text analysis, the texts are converted into probabilities and are multiplied to get the predicted classification. For instance, if the user wants to find the probability that the review "it was bad and terrible" expresses a negative sentiment. First, the total number of times 'bad' is occurred in the input text is found out, and is divided by the total number of words in the input text to get the probability of 'x' given 'y'.

$P\left(\frac{x_i}{c}\right)$ = Count of x_i in documents of class c / total no. of words in documents of class c

According to Bayes Rule, the probability of a particular document belonging in c_i is given by:

P (c_i/d) = (P (d/c_i) * P (c_i))/P (d)

The similar process is carried out for 'terrible' (the keywords 'it' and 'was' would already be removed as stop-words). It would multiply all two probabilities, and then multiply by the probability of any document expressing a negative sentiment to get the final probability that the sentence expresses negative sentiment.

The same process is followed for positive sentiment, and then whichever probability is greater would be the class that the review is assigned to.

3.2 System Design

3.2.1 Class diagram

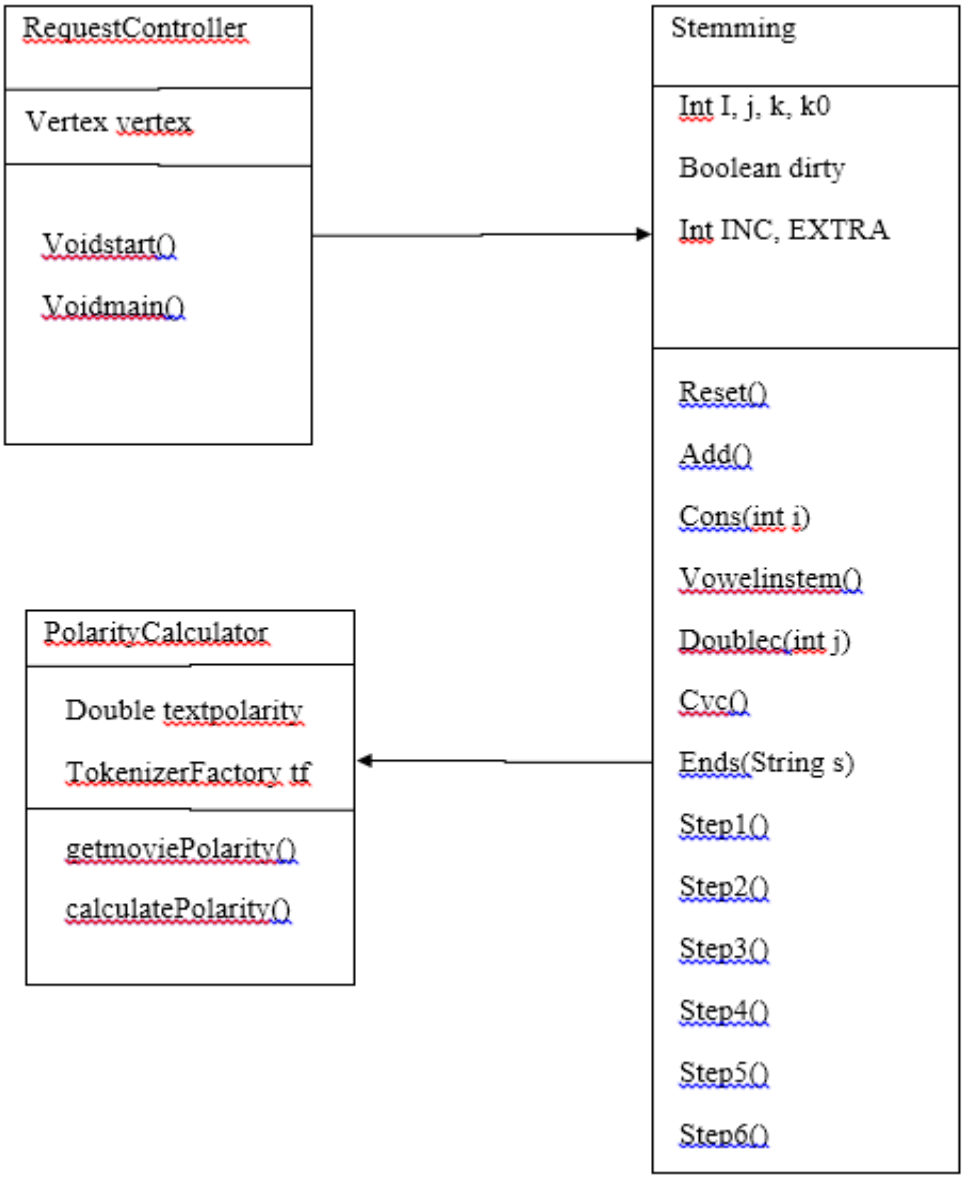


Figure 5 - Class diagram of the application

The application consists of 3 different classes. First one is RequestController. This class takes the input from the user and loads the user defined dictionary. This class uses two methods called voidstart() and voidmain(). Another class is PorterStem. It is mainly used to remove the redundant words that give the same meaning. It uses 15 methods. Another

class is PolarityCalculator. It is the class which checks the polarity of the each word provided by the user and compare with the hash table.

3.2.2 Sequence diagram

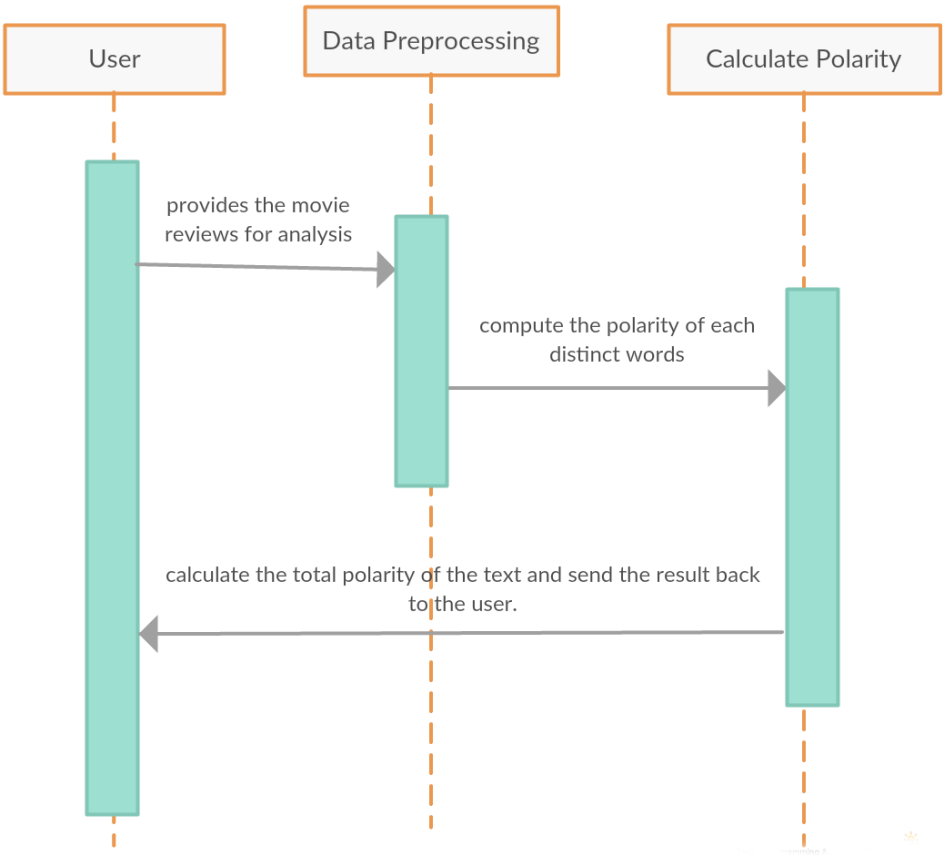


Figure 6 - Sequence diagram of the application

This Sequence diagram shows how the objects of the application operate with one another and in what order. Firstly, user provides the movie reviews as an input text to the application. That textual content goes through different data pre-processing techniques like tokenization, stop-words removal, porter stemming, etc. and lists all the available words in the input text. Those words are then sent for polarity check, in which the words are provided with different polarity values based on the hash table. Based on the polarities of individual words, the total polarity of the input text is calculated and the sentiment is detected. Then, the result is sent back to the user.

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Implementation

The main purpose of implementation is to be able to automatically classify if the input text is positive or negative. Our application consists of a list of manually classified words with its' positive and negative sentiments. Based on the sentiments, polarity of each words is calculated and are stored in hash Table.

Sample test sentences:

A) The movie is awesome. Positive.
B) Leonardo De caprio has done the best in his role. Positive.
C) The presentation was poor. Negative.
D) I didn't enjoy the movie. Negative.

```
public class RequestController extends AbstractVerticle{
   public static void main(String[] args) {
      DataDictionary.loadDictionary();
      DataDictionary.loadNegation();
      DataDictionary.loadStopWords();
      if
(DataDictionary.wordDictionary.size()==0&&DataDictionary.negationWord.size()==
0&&DataDictionary.wordsTobeIgnored.size()==0){
         System.out.println("Please View the Error Log and Restart Again");
         System.exit(0);
      }
      System.out.println("Deploying Verticle");
      Vertx vertx = Vertx.vertx();
      vertx.deployVerticle(new RequestController());
   }
```

The above-mentioned code takes the input from the user and all the individual classes that are needed to perform text analysis are loaded.

To implement the project, the main focus was to compare the polarity of the input text with a set of words that are stored in the system with their respective polarities. This project used a list of senti-words file that was provided online. The senti-words file contains a list of 83,000 words with their respective sentiments. From those sentiments, the polarity is calculated by performing (positive sentiment – negative sentiment). When the user submits the input text, the text goes through tokenization, which creates a list of tokens. The list of tokens are then pre-processed to get rid of the stop-words and stems. Those processed list of words are then compared with the file that stores the list of words with their respective polarities.

Then, the polarity of each words is calculated. With the help of that, the total polarity of the input text is calculated. If the polarity is positive, then the review is regarded as a positive review, whereas, if the polarity is negative, then the review is regarded as a negative review.

This means that if we provide the text, "Inception is an amazing movie, the plot is very good", then the system goes through the above mentioned process to determine that the review is positive.

4.1.1 Tools used

To build the project, various tools and technologies were used. Some of them are discussed below.

Java

The project is built in the java language. Using the features of Java, we have built various classes in our project. Some of the classes are StopwordRemoval, which removes the stop word from the sentences. Another class is PorterStem, it analyzes the word from the sentence and replaces the words of different verb form into unique form (verb1). It is

fully based on the object oriented paradigm. Vertex Framework was used to integrate the front-end with back-end.

Creately

Creately was used to design the sequence diagram and the use-case diagram. Creately is the diagramming and designing software which helps to design the various technical diagrams. It is a cloud-based diagram tool built on Adobe's Flex/Flash technologies and provides a visual communication platform for virtual teams. In this project, Creately was used to build UML class diagram and sequence diagram.

HTML/CSS

HTML/CSS was used to design the front end of the application. HTML stands for Hyper Text Markup Language, which adds the object in HTML document. CSS is the style sheet language that designers style the page with, to tell browsers to change the color, font, layout, and more. CSS was used to design the home page.

Idea Intellij

The project was built in Java language, so IntelliJ IDEA was used as a Java Integrated Development Environment (IDE). The whole project is written in this software.

4.1.2 Listing of major classes

RequestController

It is the class which helps to take the input text from the user.

Void start ()

This method helps to run the remote server in port 9000 and ask the input text from the user.

```
public void start() throws Exception {
        System.out.println("Verticle Deployed Successfully.");
        Router router = Router.router(vertx);
vertx.createHttpServer().requestHandler(router::accept).listen(9000);
        router.route("/").handler(rtx->{
          rtx.response().setChunked(true);
          rtx.response().sendFile("webroot/index.html");
        });      router.route("/Js/*").handler(StaticHandler.create("webroot/Js"));
router.route("/Css/*").handler(StaticHandler.create("webroot/Css"));
router.route("/images/*").handler(StaticHandler.create("webroot/images"));
```

void main()

This method loads the user defined dictionary in the project.

```
 public static void main(String[] args) {
   DataDictionary.loadDictionary();
   DataDictionary.loadNegation();
   DataDictionary.loadStopWords();
    if
(DataDictionary.wordDictionary.size()==0&&DataDictionary.negationWord.size()==0&
&DataDictionary.wordsTobeIgnored.size()==0){
        System.out.println("Please View the Error Log and Restart Again");
        System.exit(0);
    }
    System.out.println("Deploying Verticle");
    Vertx vertx = Vertx.vertx();
```

```
    vertx.deployVerticle(new RequestController());
}
```

Stemming

It is the class which helps to bring all the related words (verb) to recognizable verb form (verb-1). The method used in this class are:

Reset(): Add a character to the word being stemmed.

Add():After a word has been stemmed, it can be retrieved by toString(),or a reference to the internal buffer can be retrieved by getResultBuffer and getResultLength

Cons(int i): measures the number of consonant sequences

Vowelinstem(): find the words that ends with vowel words

Doublec(int j): find the words that ends with consonant words

Cvc(): is true if it has the form consonant - vowel – consonant and also if the second c is not w, x or y. this is used when trying to restore an e at the end of a short word.

Ends(String s): is true to the characters in the string s, readjusting

stem(): Stem a word that Returns true if the stemming process resulted in a word different from the input.

PolarityCalculator

It is the class which checks the polarity of the each word provided by the user and compare with the hash table.

getMoviePolarity

This method takes the input provided by the user and perform the text analysis.

```
public double getMoviePolarity(String inputText){
```

```
    List<String> tokenizedText = getTokenizedWords(inputText);
    double textPolarity = calculatePolarity(tokenizedText);
    return textPolarity;
}
```

getTokenizedWords

This method tokenized the text input provided by user and separate the word from the text.

```
List<Word> tokens_words = tf.getTokenizer(new StringReader(content)).tokenize();
List<String> tokenizationCompleted = new ArrayList<>();
Pattern p = Pattern.compile("[^a-z]");
tokens_words.stream()
        .filter(item-> !p.matcher(item.word()).matches())
        .collect(Collectors.toList())
        .forEach(tokens->{
            String tokenizedWord = tokens.word().trim();
            if(!tokenizedWord.trim().equals("")){
                Set<String> wordsTobeIgnored = DataDictionary.wordsTobeIgnored.keySet();
                boolean unusable = false;
                for(String word:wordsTobeIgnored){
                    if(tokenizedWord.equalsIgnoreCase(word)){
                        unusable = true;
                    }
                }
```

```
            if(!unusable){
                tokenizationCompleted.add(tokenizedWord);
            }
        }
    });
```

CalculatePolarity**()**

This method calculates the polarity of each word separated from the tokenized string and compares it with the hash table.

```
public double calculatePolarity(List<String> tokenizedWord){
    Stemmer s = new Stemmer();
    double polarity=0.0;
    int itemIndex = 0;
    for (String item:tokenizedWord) {
        System.out.println(item);
        String stemmeredWord = s.stem(item);
        System.out.println(stemmeredWord);
        if (DataDictionary.wordDictionary.get(stemmeredWord) != null) {
            double tempPolarity = DataDictionary.wordDictionary.get(stemmeredWord);
            if (itemIndex > 0) {
                String previousWord = tokenizedWord.get(itemIndex - 1);
                System.out.println("Previous Word: "+previousWord);
                Set<String> negationWords = DataDictionary.negationWord.keySet();
```

```
            for(String neg:negationWords){
               if(neg.equalsIgnoreCase(previousWord)){
                  System.out.println("Inside Negation Flag.");
                  if (tempPolarity < 0) {
                     tempPolarity = Math.abs(tempPolarity);
                  } else {
                     tempPolarity = tempPolarity * -1;
                  }
                  break;
               }
            }
         }
         System.out.println("Key Word: " + stemmeredWord + " and Polarity: " +
tempPolarity);
         polarity += tempPolarity;
      }
      itemIndex++;
```

4.2 Testing

Other than evaluating and validating the classifier's performance, different test cases were created in order to make sure that the system delivers the required output and functions properly. These test cases ensured the validity and reliability of the entire system. Unit and integration testing were performed on the system components.

4.2.1 Unit testing

Each unit of the system was tested for its correct and proper functionality.

Table 3 - Unit testing of different components

Test no.	Unit	Test	Expected Result	Test Outcome	Evidence
1	Pre-processing	Trimming delimiter symbols	Remove delimiter from input text	Successful	Test A
2	Stop word removal	Removing stop words	Remove stop words from input text	Successful	Test B
3	Tokenize	Tokenizing into sentences	Tokenizing input text into sentences	Successful	Test C
4	Tokenizer	Tokenizing into words	Tokenizing input text into words	Successful	Test D
5	Stemmer	Stemming the right words	Necessary Input word perform stemming	Successful	Test E

Test A: Delimiter successfully removed from Input text

Input**:** Films adapted from comic books have had plenty of success , whether they’re about superheroes (batman , superman , spawn) , or geared toward kids (casper) or the art house crowd (ghost world) , but there’s never really been a comic book like from hell before.

Output**:** Films adapted from comic books have had plenty of success whether there about superheroes batman superman spawn or geared toward kids casper or the art house crowd ghost world but there's never really been a comic book like from hell before

Test B: Stop Word successfully removed from Input text

Input**:** Nice movie. Way better than the tripe that the khans keep serving in regular intervals

Output**:** Nice movie. Way better tripe khans keep serving regular intervals

Test C: Tokenized input text into sentences.

Input**:** The story is not about kidnapping. It's about the feeling what people go through when such things happen. In the Oscar winning Revenant Caprio killed the one who killed his son (adopted). Here Irfan accepted the law after they realized what they had done.

Output**:** The story is not about kidnapping.

It's about the feeling what people go through when such things happen.

In the Oscar winning Revenant Caprio killed the one who killed his son (adopted).

Here Irfan accepted the law after they realized what they had done.

Test D: Tokenized input text into words.

Input**:** Absolute Pathetic movie.

Output**:** Absolute

Pathetic

movie

Test E: Porter Stemming.

Input**:** Another not so good to watch movie by talented Rajeev. After Aamir, he has hardly done any worth

Output**:** Another not so good to watch movie by talent Rajeev. After Aamir, he has hardly done any worth

4.2.2 Integration testing

For integration testing, different individual modules were combined and tested as a group. The integration testing of the system is illustrated in the table below.

Table 4 - Integration testing of the combination of different modules.

Test no.	Part	Test	Expected Result	Test Outcome	Evidence
1	Compute polarity	Give overall polarity of each word of input text	Overall polarity of text calculated	Successful	Test F
2	Classification of text	Classify input text on the basis of positive, negative & neutral	Input text classified	Successful	Test G
3	Visualization of Result of Sentiment Analysis	Display the sentiment expressed by input text	Sentiment expressed by the input text are expressed	Successful	Test H

Test F: Compute Polarity

Input**:** The movie was as good as expected.

Output**:** Overall sentiment Score: +1.0

Test G: Classification of Text

Input**:** It's about the feeling what people go through when such things happen. In the Oscar winning Revenant Caprio killed the one who killed his son (adopted). Here Irfan accepted the law after they realized what they had done.

Output**:** Positive sentiment Score: +2.68

Negative sentiment Score: -4.73

Neutral sentiment Score: 0.95

Test H: Visualization of Result

Input**:** The movie was awesome.

Output**:** The Review is Positive.

CHAPTER 5: MAINTENANCE AND SUPPORT

5.1 Corrective Maintenance

This application will be hosted and once the user reports of a bug, the bug will be fixed right away.

5.2 Perfective Maintenance

The performance of the system will be improved constantly. The performance problem will be identified and fixed. The application will be improved constantly so that it will use minimum resources and gives maximum performance.

5.3 Preventive Maintenance

The possible errors that can occur in future will be predicted and the system the possible strategy to cope with the future error will be implemented.

CHAPTER 6: CONCLUSION AND RECOMMENDATION

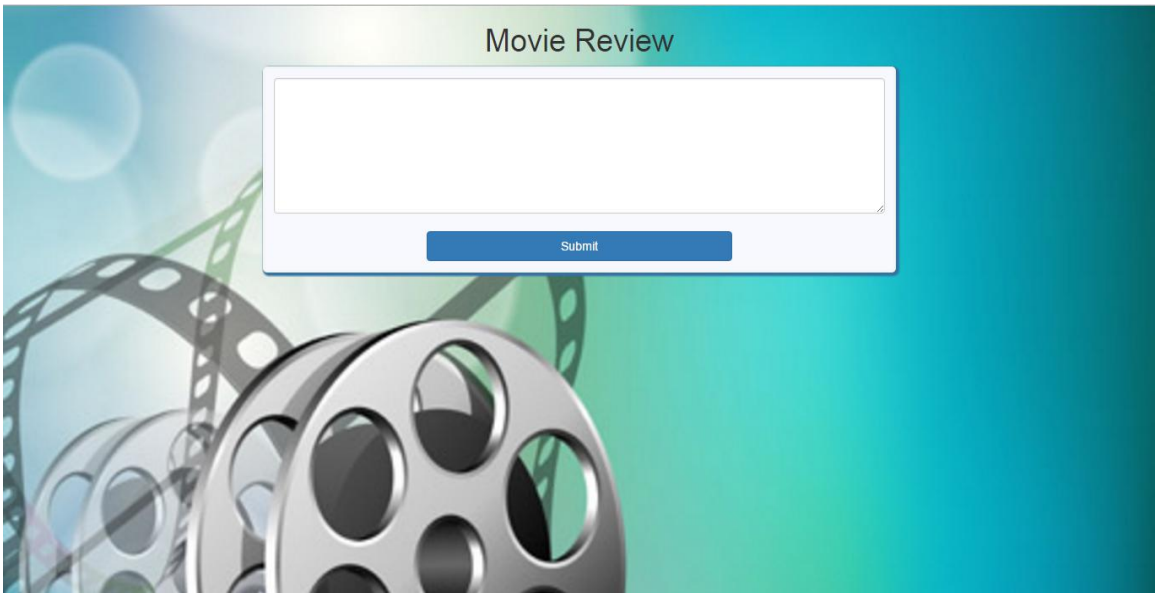
6.1 Conclusion

In this project, we have used Baseline approach to analyze the reviews. The project serves the result of analyzed movie reviews to the users. We focused on processing the texts to extract valuable information from the movie review to provide the user with the result if the movie is good or bad. In delivering a quality solution, we conducted early tests on the application prototype, and resolved the early issues.

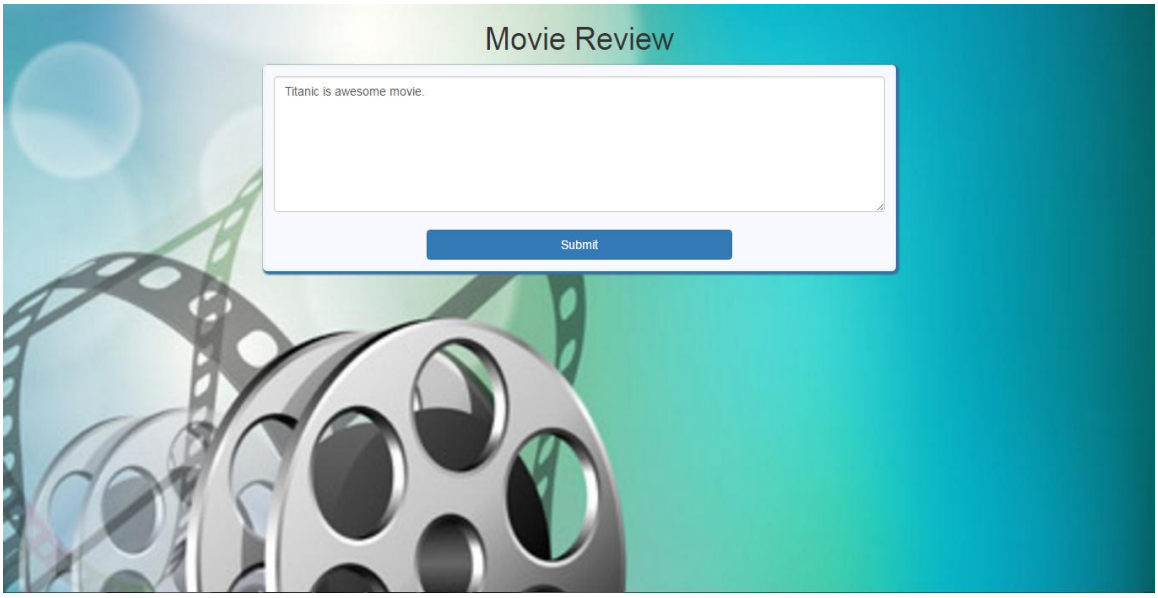
6.2 Recommendation

This application can be further enhanced to visualize the results and give real time updates on the movies that are currently running on theatres.

APPENDIX

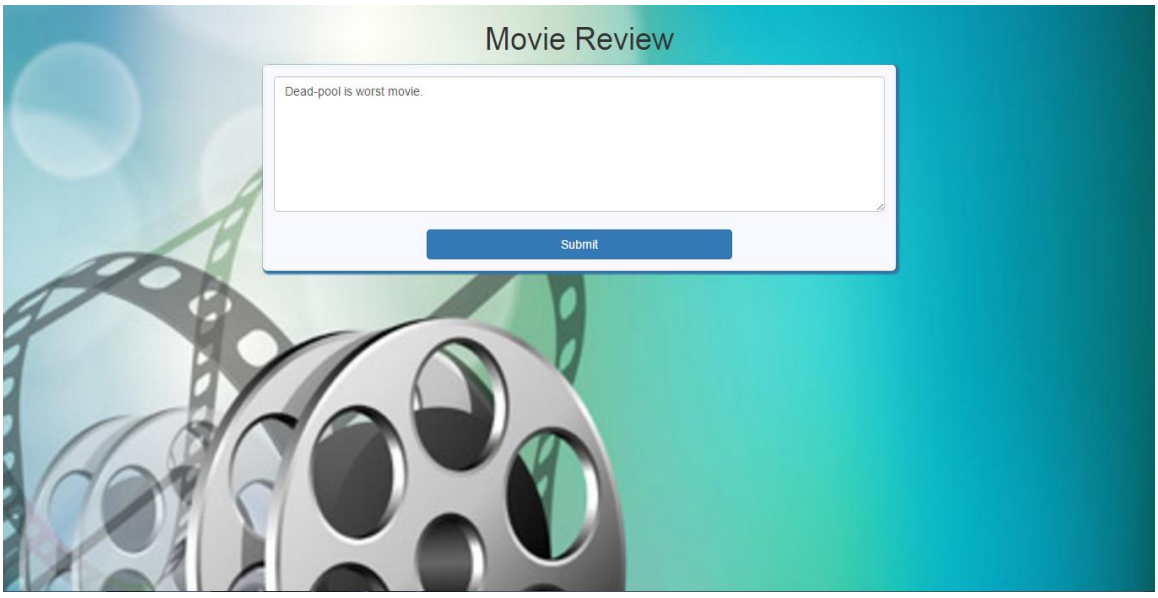


The initial home page of the application.



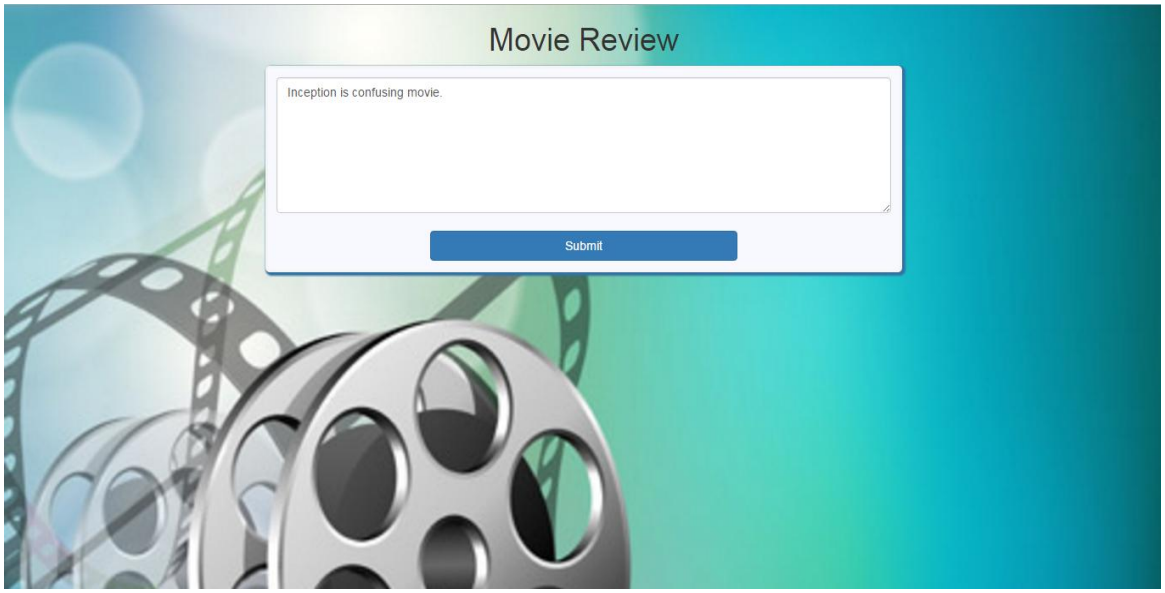
Inputting the text on the text area.

The result of the input text displayed back to the user as ‘positive review’.

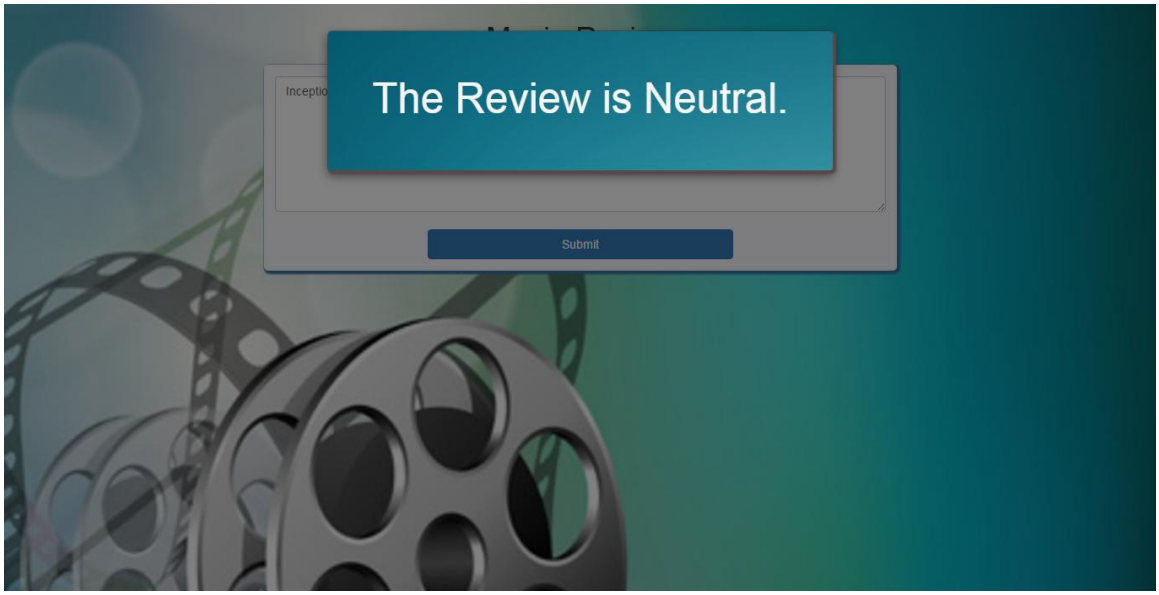


User inputting negative input text on the text area.

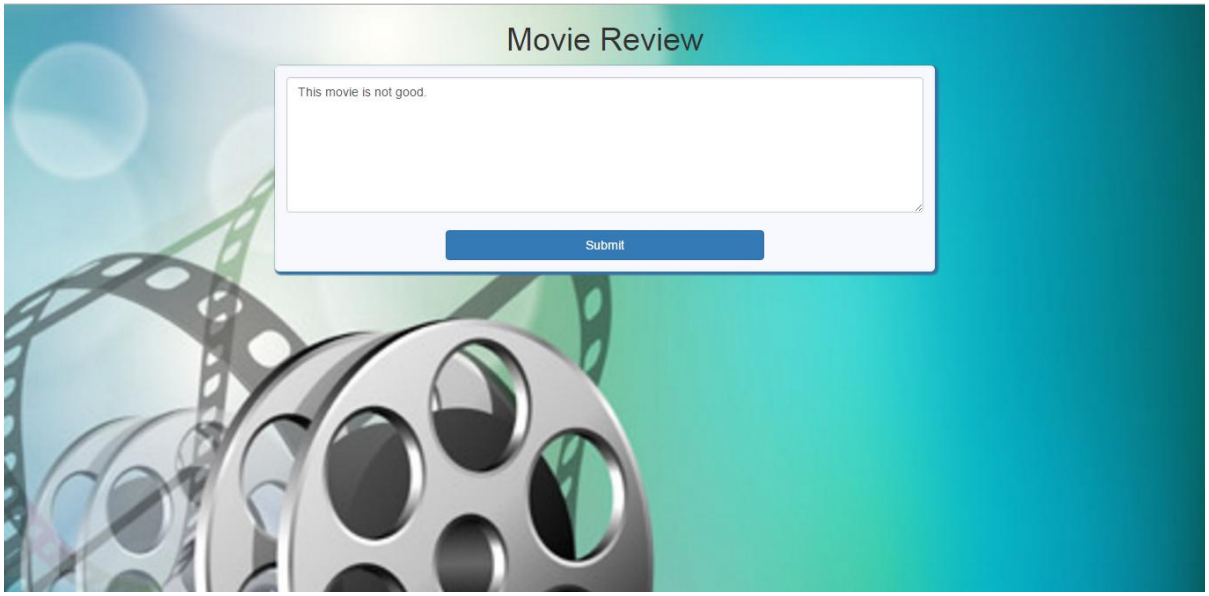
Result of the input text displayed back to the user as ‘negative review’.



User inputting neutral text to the text area.



Result of the input text displayed back to the user as ‘neutral review’.



User inputting the text using negations.

Result displayed to the user as ‘negative review’ by processing negation handling technique.

REFERENCES

[1] Vivek Narayanan, Ishan Arora, Arjun Bhatia, Fast and Accurate Sentiment Analysis using an enhanced Naïve Bayes Model, 2013.

[2] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011, June). Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics.

[3] Bo Pang and Lillian Lee, Opinion Mining and Sentiment Analysis, 2008.

[4] Bo Pang and Lillian Lee, "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts", Proceedings of the ACL, 2004

[5] Rajaraman, A.; Ullman, J. D. (2011). "Data Mining". Mining of Massive Datasets (PDF).

[6] Ribeiro, Filipe Nunes; Araujo, Matheus, "A Benchmark Comparison of State-of-the-Practice Sentiment Analysis Methods", 2010.

[7] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts, Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[8] Devitt, Ann, and Khurshid Ahmad. "Sentiment polarity identification in financial news: A cohesion-based approach", 2007.

[9] Godbole, N., Srinivasaiah, M., Skiena, S.: Large-scale sentiment analysis for news and blogs (demonstration).In: Proc. Int. Conf. Weblogs and Social Media (ICWSM 07). (2007)

[10] C. P. Gupta and B. K. Bal, "Detecting Sentiment in Nepali Texts: A bootstrap approach for Sentiment Analysis of texts," in Cognitive Computing and Information Processing (CCIP), 2015.

[11] Timmaraju Aditya, Khanna Vikesh, Sentiment Analysis on Movie Reviews using Recursive and Recurrent Neural Network Architectures, 2015.