[Chapter 5](#_Toc265326415)

Implementation

An implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system. The low level design details are converted into a language specific program such that they satisfy the requirements of the given software. The technique used for implementing the software must support reusability, ease of maintenance and should be well documented. The choice for implementation also plays a very important role. The implementation phase should follow a well-structured approach which is based on popular software engineering practices.

The implementation phase of any project development is its most important phase as it yields the final solution, which solves the problem at hand. The implementation phase involves the actual materialization of the ideas, which are expressed in the analysis document and developed in the design phase. Implementation should be a perfect mapping of the design document in a suitable programming language in order to achieve the necessary final product. Often the product is ruined due to incorrect language choice for implementation or an unsuitable method for programming. The factors concerning the programming language and platform chosen are described in the next couple of sections.

[**5.1.** **Programming Language Selection**](#_Toc265326418)

The programming language chosen must reflect the necessities of the project to be completely expressed in terms of the analysis and the design documents. Therefore before choosing the language, features to be included in the project are decided. The Sentiment Analyzer project needs the following features in a language to be implemented. Some of the features required are stated as follows:

* J2EE provides us with servlets and JSP which help in dynamically constructing web pages.
* JSP and servlets make use of Java backend in a very optimal manner. They have special tags which help us exploit these features.
* J2EE provides us with Java Beans which help in proper data manipulation.
* Java's core classes are designed from scratch to meet the requirements of an object oriented system.

[5.2. Platform](#_Toc265326419)

The Sentiment Analyzer tool was built and designed on Windows Operating system family. They were specifically tested on Windows 7 with Google Chrome and Mozilla Firefox browsers. Because the product is browser based, any user with the browsers mentioned above will be able to run the tool. The product is hence platform independent in the true sense.

[5.3. Code Conventions](#_Toc265326420)

The code standards for the Java programming Language document contain the standard conventions that follow. It includes file names, file organizations, indentation, comments, declarations, naming conventions and programming practices. Code conventions improve the readability of the software.

**5.3.1. Naming Conventions**

A naming convention is a set of rules for choosing the character sequence to be used for identifiers which denote variables, types, functions, and other entities in source code and documentation. There are several common elements that influence most if not all naming conventions in common use today. They are:

* Use mixed case to make names readable
* Avoid long names (15 characters maximum is a good idea)
* Avoid names that are too similar or that differ only in case
* Capitalize the first letter of standard acronyms
* Use terminology applicable to the domain
* Use full descriptors that accurately describe the variable, field, or class

**5.3.2. File Organization**

As stated above, this project has been developed using the eclipse JEE IDE. This is a dynamic web project. The project has been organized as follows:

* Java Resources - This folder contains the java classes organized in packages. All the algorithms implemented in java are present in this folder under different packages.
* JavaScript Resources - This folder contains the JavaScript library files.
* Web Content - This folder mainly contains the jsp, js, html, css files which are used for developing the front webpages.

**5.3.3. Comments**

* Implementation Comment Formats

- Block Comments

/\*

\* Here is a block comment.

\*/

-Single Line Comments

if (condition) {

/\* Handle the condition. \*/

...

}

-Trailing Comments

if (a == 2) {

return TRUE; /\* special case \*/

}

else {

return isPrime(a); /\* works only for odd a \*/

}

-End-of-Line Comments

if (foo > 1) {

// Do a double-flip. Harlem Style

...

}

else {

return false; // Explain .

}

* Documentation Comments

/\*\*

\* The Example class provides ...

\*/

public class Example { ...

**5.4. Difficulties Encountered and the Strategies Used To Tackle**

There were a number of challenges that were faced while implementing the Sentiment Analyzer tool. Some challenges were tricky and ended up in helping us think innovatively and come up with efficient solutions. Some major problems that were encountered have been stated in brief along with their solutions.

**Problem 1**

Initially we were using double data type to hold the intermediate and final computed probability values, but the probability values were so small that in some cases they ended being zero.

**Solution**

We used log function on the intermediate and final computed probability values, hence the values obtained were not close to zero.

**Problem 2**

In the initial stages of the project charts4j libraries were used to plot graphs and pi charts. There were some internal problems with the URL rendering.

**Solution**

This Problem was solved later by making use of Google's Charts API and

Java script.

**5.5 Module Design**

Object-oriented programming (OOP) is a programming paradigm that represents concepts as "objects" that have data fields (attributes that describe the object) and associated procedures known as methods. Objects, which are instances of classes, are used to inter- act with one another to design applications and computer programs.

Had it not been for the presence of the OOP paradigm, our efforts in this project would have gone in vain, and we do not use that term lightly. Code management was a whole lot easier when compared to our past experience with procedural programming. In this chapter, we describe the different packages that were created by us to efficiently manage our code. Figure 5.1 shows the UML class diagram of the application. Know first that our application consists of one diverse package, namely:

Tweetification

It consists of following classes:

1. TweetSentiment.class
2. ReadTrainingData.class
3. StartProgram.class
4. NegativeProbability.class
5. PositiveProbability.class
6. NuetralProbability.class
7. Replacer.class
8. WordCounter.class
9. ComputeLarge.class

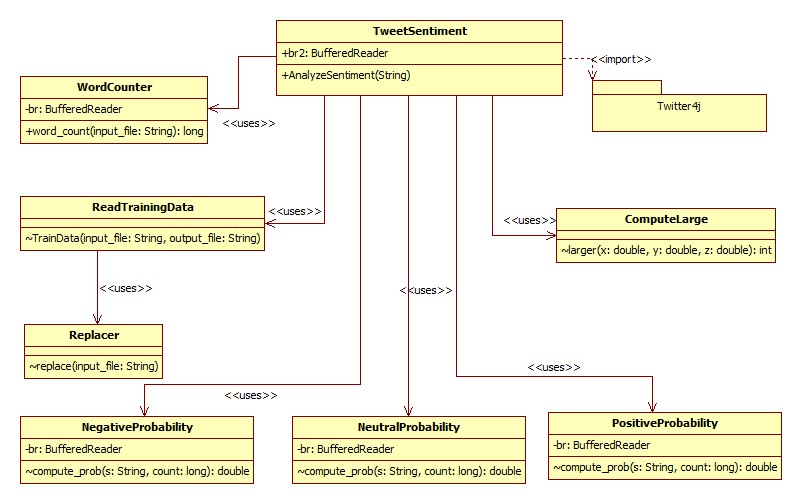


Figure 5.1 UML class diagram

**5.5.1 TweetSentiment .class**

This class uses Twitter4j API to access the tweets from Twitter. In AnalyzeSentiment method tweets are stored in a text file. It will replace all the ambiguities, Train the model and return Integer array that consists of tweet counts, positive sentiments, negative sentiments, neutral sentiments, others.

**5.5.2 ReadTrainigData.class**

This class contains a method called trainData in which we are passing input file and output file. It simply reads the input file and replaces the ambiguities and writes it to output file.

**5.5.3 StartProgram.class**

This class reads data from two files. i.e., one file that contains tweets and second one is a result of machine learning algorithm. It correlates the tweet with corresponding result.

**5.5.4 NegativeProbablity.class**

It consists of compute\_prob method which takes twitter file and count of the words that are present in training file. Naïve Bayes machine learning technique is implemented here. It will return the computed probability.

**5.5.4 PositiveProbablity.class**

It consists of compute\_prob method which takes twitter file and count of the words that are present in training file. Naïve Bayes machine learning technique is implemented here. It will return the computed probability.

**5.5.4 NeutralProbablity.class**

It consists of compute\_prob method which takes twitter file and count of the words that are present in training file. Naïve Bayes machine learning technique is implemented here. It will return the computed probability.

**5.5.5 Replacer.class**

It does following functionality

1. Usernames Users often include Twitter usernames in their tweets in order to direct their messages. A de facto standard is to include the @ symbol before the username (e.g. @alecmgo). An equivalence class token (USERNAME) replaces all words that start with the @ symbol.
2. Usage of links Users very often includes links in their tweets. An equivalence class is used for all URLs. That is, we convert a URL like \http://tinyurl.com/cvvg9a" to the token \URL."
3. Tweets contain very casual language. Example, if you search \hungry" with an arbitrary number of u's in the middle (e.g. huuuungry, huuuuuuungry, huuuuuuuuuungry) on Twitter, there will most likely be an on empty result set. We use preprocessing so that any letter occurring more than two times in a row is replaced with two occurrences.

**5.5.6 WordCounter .class**

It computes total number of words present in a given file.

**5.5.7 ComputeLarge.class**

It computes larger value for given two numbers.