THE UNIVERSITY OF ZAMBIA

SCHOOL OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER STUDIES

SOCIAL MEDIA INTELLIGENCE EXTRACTOR

By

Mutanuka Kasonde

Chilembo Musafili

Chulu Victor

A thesis submitted in partial fulfillment of the requirement for the degree of Bachelor of Computer Science in the Department of Computer Studies.

SUPERVISOR: Mr. S. Mwanza

Check key words

ABSTRACT

In the last few centuries, information has highly appreciated in value and its sources have greatly increased in number. The introduction of the web 2.0, which allowed users not only to retrieve but also generate information as authors, gave birth to a whole new data source called *social media*. Social media is a term that integrates technology, social interaction and user generated content and differs from traditional broadcasting. It commonly comprises of technologies such as instant messaging programs, discussion forums, weblogs and wikis. Facebook, YouTube and Wikipedia are examples of popular social media websites. The information generated is greatly useful to individuals, organizations, institutions and governments globally. But this information tends to be highly unstructured and certain times not trustworthy.

Social Computing is a novel and emerging computing paradigm that involves a multi-disciplinary approach in analyzing and modeling social behaviors on different media and platforms to produce intelligent and interactive applications and results. The objective of this project is to summarize social media opinions on various subjects with the focus on Twitter and Facebook microblog systems. In this project, we propose and attempt to implement a system that uses various machine learning and computational techniques used in Social Computing to collect, extract, process, mine, and visualize the data. This summarization task is different from traditional text summarization because we are only interested in the positive, negative and neutral opinions people have expressed on specific features or topics. This will be done at both the sentence and at the post level (document level).

Keywords: Social media, web crawling, text processing, data mining, machine learning, topic detection, sentiment analysis, quality content detection.

DECLARATION

We, the undersigned here declare that the *Social Media Intelligence Extractor* is our own work, that it has not been submitted for any degree or examination in any other university and that all the sources we have used or quoted have been indicated and acknowledged by complete references.

Authors:

Mutanuka Kasonde	Chilembo Musafili	Chulu Victor
10060413	10108475	10019022
.....	.....	.....

July 2014

Supervisor:

Mr. Mwanza Selvas

.....

DEDICATION

To my beloved parents Maj. D. Mutanuka and Mrs. M. C. Mutanuka for their love, support, patience and belief in me. I would not have the goals I have to endeavor and be the best to reach my dreams. I wish to thank you for giving me a chance to substantiate and improve myself through all walks of life. I love you

- Mutanuka Kasonde

To the memory of my only two brothers, Bright Kamata N'gombe Chilembo and Aubrey Zondani Chilembo, both present at beginning of my degree but absent at the end.

- Chilembo Musafili

To my friends and family, without whom I would not be who I am today.

- Chulu Victor

ACKNOWLEDGMENT

First of all, we would like to thank the Lord God almighty for his loving nature. Secondly, we would also like to thank our supervisor; Mr. Mwanza Selvas who through his guidance and support helped us accomplish this project. We would also like to thank all the members of staff of the Department of Computer Studies, who in one way or another have helped us become who we are.

CONTENTS

ABSTRACT i
DECLARATION ii
DEDICATION iii
ACKNOWLEDGMENT iv
1.0 INTRODUCTION 1
1.1 Problem statement 1
1.2 Project Objectives 1
1.3 Role of the Students in project 3
1.4 Structure of Thesis 4
1.5 Scope 5
1.6 Expected Benefits 6
2.0 LITERATURE REVIEW AND RESEARCH METHODOLOGY 7
2.1 Literature reviews on topics related to the project 7
2.2 Research methodology 10
2.3 Review on several software development methodology 11
2.4 Review of current systems 13
2.5 Review of similar systems 14
3.0 REQUIREMENT ANALYSIS 15
3.1 Software development methodology 15
3.2 Surveys and its results 15
3.3 System goals analysis 18
3.4 System functionalities 19
3.4 Development tools and software 31
4.0 SYSTEM DESIGN 33
4.1 Explanation of the proposed 33
4.2 Software level Architectural Design 34
4.3 Modular design of the System functions 35

4.4 System structure Data Flow Diagrams and Entity Relation Diagram 38
4.5 User Interface Design 44
4.6 System interface design 45
5.0 SYSTEM IMPLEMENTATION 47
5.1 Technical details of the developed system and screen shots 47
5.1.1 Data Collection Module 47
5.1.2 Opinion Extractor 52
5.1.3 Data Preprocessing Module 56
5.1.4 Feature Finder Module 59
5.1.5 Tweet Sentiment Classifier Module 65
5.1.6 Polarity Discovery Module 68
5.1.7 Storage Module 76
5.1.8 Presentation Module 77
6.0 TESTING AND VERIFICATION 86
6.1 System test method 86
6.2 Data collection testing 86
6.3 Data analysis testing 89
6.4 Ease of use testing 90
6.5 Aesthetics testing 91
7.0 CONCLUSION 92
Other Achievements 92
Challenges and Recommendations 93
8.0 REFERENCES 95
APPENDIX A 98
APPENDIX B 99

1.0 INTRODUCTION

1.1 Problem statement

Social media like Facebook and Twitter have become so popular in recent years attracting attention from individuals, organizations and governments. Many people use it for communication and also as a platform for expressing opinions on different subjects like politics, sports, products etc. Because the data produced by social sites is open to the public, most organizations and governments are now using this data to monitor people's reactions to their products, services, policies, and so on. Because social media attracts people from different parts of the world, monitoring its data gives organizations and governments a global view of the matter under consideration.

However, the amount of data produced by Social Sites is huge and unstructured (i.e. contains a lot of grammatical and spelling errors and contains a lot of slang), which makes the process of monitoring peoples' opinions on different subjects difficult.

In this project, we design a software system that summarizes peoples' opinions expressed on different topics. The system extracts topics from social media data and mine positive, negative and neutral opinions expressed by people. Because social media provides a live stream of its data, the system will allow organizations and governments to monitor public opinions toward different topics easily and quickly.

1.2 Project Objectives

In order to achieve and implement the above proposed system, the following will be the components that it will constitute. These components define the primary objectives of the project as they are itemized below;

1. Data collection:

Social media data will be collected from Twitter and Facebook.

2. Data Preprocessing:

   Social media data is highly unstructured and contains a lot of grammatical and spelling errors. The data will be checked and corrected on spelling and grammar.

3. Topic Detection:

   This involves finding out what topic someone is talking about in a post. People comment on physical object (like products, places, etc.) and nonphysical objects (ideas, concepts, etc.). Topic detection should be able to handle both kinds of objects being commented on. Objects have attributes/features and behavior. For example, a phone has a screen, a battery which is its features, and it has behavior, e.g. it can call, it can charge and so on. Topic detection should be able to detect object features and behavior as well.

4. Sentiment Analysis:

   People like expressing opinions on objects they comment on in social media. Sentiment analysis involves detecting negative, positive and neutral opinions expressed on different topics. Sentiment must be detected on the social media post level and on the topic level.

5. Quality Content Detection:

   The quality of user-generated content varies drastically from excellent to abuse and spam. Quality Content Detection involves detecting the quality of social media post. This can be achieved by analyzing social media properties like the number of likes and so on and so forth. People have accounts in different social sites. Optionally, we can try to detect accounts in different social sites that belong to the same person. This data can be used to measure the quality of messages someone posts.

6. Product Feature – Opinion summarized reporting:

For each product feature extracted, the system will show the number of microblog messages that express positive, negative and neutral opinion. An example of the output is given below;

Android

Screen:

Positive: 253 <individual microblogs>

Negative: 6 <individual microblogs >

Neutral: 1000 <individual microblogs >

Flexibility:

Positive: 134 <individual microblogs >

Negative: 10 <individual microblogs >

Neutral: 500 <individual microblogs >

...

1.3 Role of the Students in project

The following is the distribution of project tasks;

- Data collection: Everyone.
- Data preprocessing: Everyone.
- Topic detection will be done by Chilembo Musafili.
- Sentiment Analysis will be done by Chulu Victor.
- And Quality Content Detection will be done by Mutanuka Kasonde.
- Product Feature-opinion summarized reporting: Everyone.

1.4 Structure of Thesis

Chapter 1 Introduction: Delivers an overview of the project and provides background knowledge of the project domain and highlights the challenges of the domain and how the project serves as a solution. This chapter introduces the scope of the project and further highlights the primary objectives. Seen as how this project is a work of three students, the chapter clearly states the role of each student in this project.

Chapter 2 Literature Review and Research Methodology: Provides a synopsis analysis of works related to this thesis. For each work, the thesis shows the techniques and methodologies developed. Each work given is linked to the references for more information.

Chapter 3 Requirement Analysis: Presents the goal analysis model which breaks the high-level goals into sub-goals and requirements of this project in fine points. By this approach, you understand how high-level goals are achieved as you go down the model. The chapter further presents functional and non-functional requirements of this project. These requirements are derived from the goals of the project.

Chapter 4 System design: Offers a sketch of the all-inclusive system design. The chapter shows how the system interacts with other systems in its environment. The chapter correspondingly shows the high level system architecture which displays how the system is arranged and how it communicates to perform its functions. The chapter then shows how the system is decomposed into individual sub-systems and how these subsystems implements the requirements of the project.

Chapter 5 Implementation: This chapter gives a detailed description of the development of the system and how each sub-system of a system is designed to perform its functions. It furthermore incorporates screenshots of a developed system.

1.6 Expected Benefits

The successful implementation of this project will give birth to a whole new meaningful source of information. As it stands, there is a lot of useful information on the web but because it is highly unstructured, it remains unused. This project will provide information to individuals, organizations and governments that will help in decision making processes and market research. Below is a concise list of expected benefit

- Reduce on costs spent on market research
- Provides a wider sources of information (globe)
- Improve the speed of decision making since the information will be readily available.

2.0 LITERATURE REVIEW AND RESEARCH METHODOLOGY

2.1 Literature reviews on topics related to the project

In this section a review of previous work related to this research project is given. Research about Product Feature extraction related issues, tweet sentiment analysis and opinion mining and quality content detection are discussed.

2.1.1 Product Feature Extraction

Product feature extraction is the process of identifying features of products people have mentioned in text. They are a good number of previous works which look in this problem. Some works uses word relationships in the text. Others use other text statistics to find features. Some argue that true product features can be gotten from the manufactures. The work of [2] establishes that product features must be gotten from customers because customers use words not used by manufacturers and customers can mention features that the manufactures never thought of.

The authors of [2] extract product features by using Apriori algorithm. The Apriori algorithm works in two phases. In the first phase it finds frequent item sets in the dataset. The second phase is to find relationships in the extracted item sets. They just use the first phase of the Apriori algorithm to produce frequent nouns in the dataset. These nouns are taken as candidate features. Then they prune features using compactness and redundancy techniques to get rid of uninteresting features. Red Opal system in [8] also uses frequent noun and noun phrases for product feature extraction. Both of these solutions use well-formed and structured text.

Ontology engineering has shown that written text is a source for extracting semantic relationships like hyponyms and meronyms. Meronyms show the "has-a" relationship between two concepts. A concept X is a meronym of a concept Y if native English speakers can accept sentences formed like Y contains X or X belongs to Y. In [1], the authors study how meronyms and hyponyms can be extracted from text. This solution works on well-structured text.

2.1.2 Opinion Mining

Opinion mining is the process of discovering opinions expressed by users in text. These opinions can be positive, negative or neutral. Usually people use adjective words to express opinions. Words like *love*, *like* and so on. They are a number of works which extract opinions in text. In [1, 2, 6] opinions are extracted in online product reviews. Other systems that extract opinions in text include recommendation systems like in [7] and business intelligence in [4].

One approach to opinion mining is to use a seed of opinion words and their polarity orientation and then expand the lexicon using WordNet [1]. The second method in [5] is WordNet expansion and statistical estimation such as point wise mutual information (PMI). These are the two most popular methods. Both of these methods work on well-structured text and do not consider negation words near opinion words. The second approach is to judge the polarity of a word or sentence using a trained classifier such as Naïve Bayes and SVM.

2.1.3 Sentiment Classification

Sentiment analysis is a growing area of Natural Language Processing with research ranging from document level classification [9] to learning the polarity of words and phrases [10, 11]. Given the character limitations on tweets, classifying the sentiment of Twitter/Facebook messages is most similar to sentence-level sentiment analysis [10, 12]; however, the informal and specialized language used in tweets and posts, as well as the very nature of the microblogging domain make Twitter/Facebook sentiment analysis a very different task. It's an open question how well the features and techniques used on more well-formed data will transfer to the microblogging domain. Just in the past year there have been a number of papers looking at Twitter sentiment and buzz [4, 13, 14, 15, 16, 17, 29]. Other researchers have begun to explore the use of part-of-speech features but results remain mixed. Features common to microblogging (e.g., emoticons) are also common, but there has been little investigation into the usefulness of existing sentiment resources developed on non-microblogging data. Researchers have also begun to investigate various ways of automatically collecting training data. Several researchers rely on emoticons for defining their training data [14, 29]. [13] exploit existing Twitter sentiment sites for collecting

training data. [15] use hashtags for creating training data, but they limit their experiments to sentiment/non-sentiment classification, rather than 3-way polarity classification [18], as we do.

2.1.4 Quality Content Detection

The Assessment of content quality in social media possess a challenge in accurately evaluating the quality because the content is usually created by users from different backgrounds, for different domains and consumed by users with different requirements. Due to the large population of social media users and vast volume of user generated content in many social media services, it would be useful to automate the process of Quality Content detection over manual evaluation techniques. The ability to develop an accurate and objective measure for quality becomes challenging when one takes into consideration the diversity of user requirements apparent in most social media services. Different paradigms would require considering different techniques and dimensions which may include user feedback, reputation, reliability, accuracy, objectivity, the amount of data and many others [19, 20, 22] . i e Detecting quality in a forum may be different from a question and answer environment. In this project, we however concentrate on weblogs.

[25] Proposed a model that was able to classify weblog articles into two categories namely, informative and affective weblog articles. Informative articles have content similar to news websites, provide technical descriptions (e.g. programming techniques), common sense knowledge and/or present objective comments about events in the world. Adversely, affective articles comprise of content similar to personal diaries and can contain information relating to the weblog author's feelings and emotions on certain subjects.

In the real world, when we are faced with a situation where a decision is to be made regarding the trustworthiness of a person or a social object, we assess it based on numerous factors including the past, present and perceived qualities. The same intuition is applied here. We can derive numerous suitable features about the content (primary target of trust) depicting reputation (past), performance (present) and appearance (perceived) from the external characteristics of the

content as well as the associated metadata (secondary targets of trust). These features can together help predict the trustworthiness of content.

In this project, one of the dimensions we will explore is the reputation of the creator (user). This will be achieved by analyzing his/her past actions on the social media portal such as content creation or consumption, responses to content (this would mean considering the number or likes or re-tweets), generic interactions with others, functions such as social networking and other activities on the site. Platforms like Facebook keep a user activity log which is public and can be accessed on the timeline with permission.

We will also take into consideration the user's status. By this, we will explore the user's personality, status and identity and by so doing, we will consider the context in which the opinion is made and match it against the status, level of competence (if available) and expertise of the user in question. Surely a known president should be ranked higher on issues of the nation that a known computer science student. In a similar vein, outwardly visible characteristics of content, such as style, size and structure, may also prove useful in judging its quality. For example, the length of a blog post or article could indicate the seriousness of the author or the comprehensiveness of the article and the structure and language of the content could indicate its quality.

2.2 Research methodology

This project is not developed for a specific stakeholder. However, it would be wise to mention that not one particular method was employed but a combination of a few other methods. Through extended review of literature written on previous systems similar to respective components make up our system, we gained an insight on what works better and what does not. The process of experimentation proved to be the most useful in this project, by experimenting on different platforms and pieces of technologies, we were able to make informed discussion. With the evolution of social media, it does not escape our notice that user s from different parts of the globe have adopted certain messaging patterns which may be considered to be the web de facto standard. Hence, to achieve our primary objective and gain a better understanding of the

patterns, we employed ethnology as a research methodology with respect to their (users) interaction on media platforms.

2.3 Review on several software development methodology

2.3.1 The waterfall model

The waterfall model, one of the oldest paradigms for software engineering is sometimes called the classic life cycle. It suggests a systematic sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction and deployment, culminating in on-going support of the completed software.

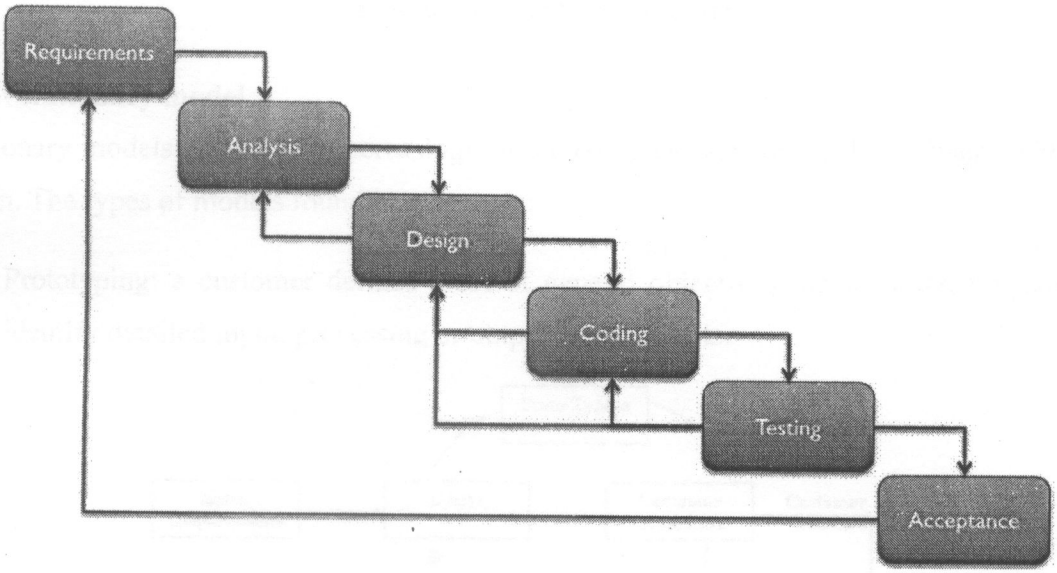

Figure 2.1: Waterfall Model

2.3.2 Incremental model

The incremental model combines the elements of the waterfall model applied in an iterative fashion. This model delivers a series of releases called increments that provide progressively more functionality for the customer as each increment is delivered.

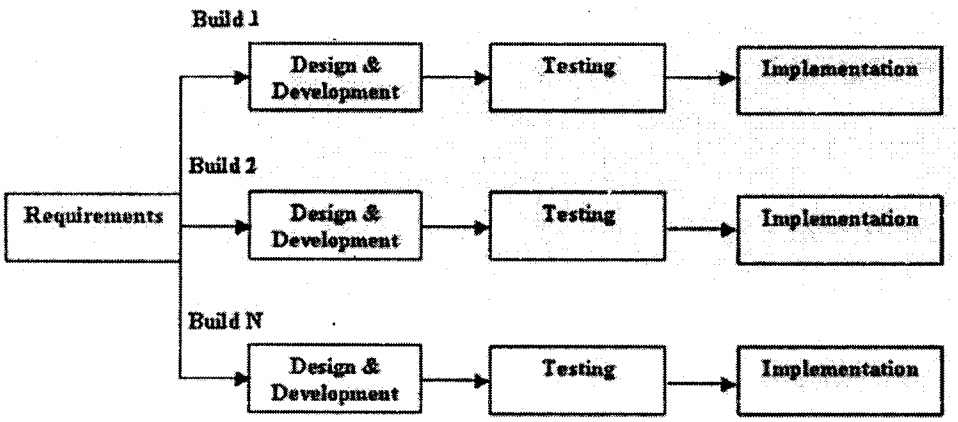

Figure 2.2: Incremental Model

2.3.3 Evolutionary model

Evolutionary models produce an increasingly more complete version of the software with each iteration. The types of models found here include:

- Prototyping: a customer defines a set of general objectives for software, but does not identify detailed input, processing or output requirements.

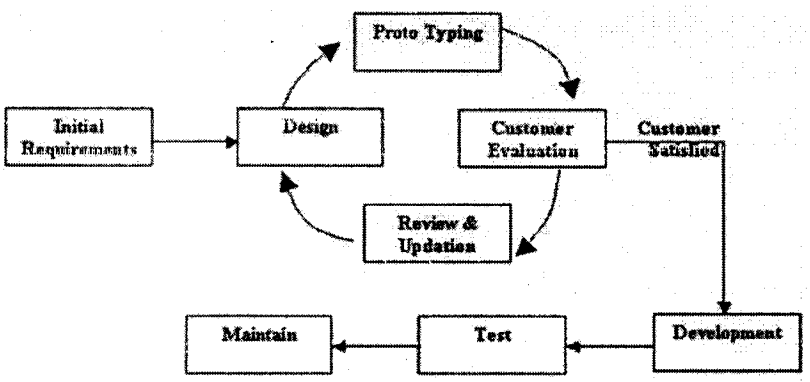

Figure 2.3: Prototyping Model

- The spiral model: couples the iterative nature of prototyping with controlled and systematic aspects of the waterfall model.

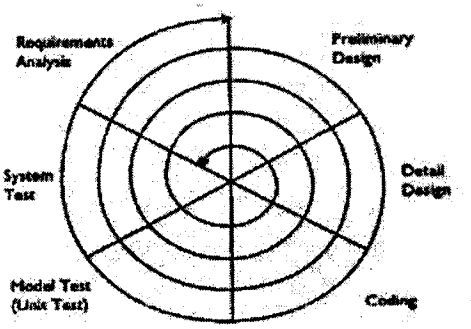

Figure 2.4: Spiral Model

- The concurrent development model: represented as a series of framework activities, software engineering actions and tasks, and their states.

2.4 Review of current systems

Considering the aims and objective of this project with regards to the problem statement, there is no currently running system to the best of our knowledge. Social media data still remains unused and unstructured. There are only a few components that stand alone and hence do not achieve what we propose.

2.5 Review of similar systems

There may be no currently installed or running systems as stated above. However, that does not imply that there are no similar works in progress. Research in the area of opinion mining using natural language processing is active. A similar system implemented is the social media marketing insight extractor. The research targeted twitter as the platform of choice .This was however designed as a desktop system supporting a single user. In this project, we attempt to implement a web application with the ability to support multiple clients.

3.0 REQUIREMENT ANALYSIS

3.1 Software development methodology

From the evolutionary models, awareness is raised that software first versions normally lead to the need of redoing the application, or some of the modules. So the first version of the software is a trial of the software that will then be confronted with the input and the revision of requirements from the users. Then the real development begins with the more solid foundations that were provided during the use of the software. The first version is also called the thrown away prototype. This method is also often called prototyping. It advocates the principle of evolution by operational experience. The model may be confused with the code and fix or ad-hoc methodology, but the discipline of the method is provided by the method to obtain the software. It can be used as an iterative process or linear one but through each prototype. It's theme word is "do it twice" and it was stated by its developer Tom Gilb [27]. The process of requirements analysis is then mixed with the development itself, when the requirements are refined and obtained from the use of the prototypes. Also maintenance and changes are almost dissimulated during the development as the development is being done during those tasks.

There are three main moments in the evolutionary model (without the discipline of the method chosen for the development) and these are:

1. Deliver a prototype for the user to experiment;
2. Measure the input of the user and the operational value of the software (in all critical aspects and qualities)
3. Adjust both design and goals according to the input taken from the real use of the software.

3.2 Surveys and its results

The Prototyping Model was developed on the assumption that it is often difficult to know all of your requirements at the beginning of a project. Typically, users know many of the objectives that they wish to address with a system, but they do not know all the nuances of the data, nor do they know the details of the system features and capabilities. The Prototyping Model allows for

these conditions, and offers a development approach that yields results without first requiring all information up-front.

When using the Prototyping Model, the developer builds a simplified version of the proposed system and presents it to the customer for consideration as part of the development process. The customer in turn provides feedback to the developer, who goes back to refine the system requirements to incorporate the additional information. Often, the prototype code is thrown away and entirely new programs are developed once requirements are identified.

There are a few different approaches that may be followed when using the prototyping model:

- Creation of the major user interfaces without any substantive coding in the background in order to give the users a "feel" for what the system will look like,
- Development of an abbreviated version of the system that performs a limited subset of functions; development of a paper system (depicting proposed screens, reports, relationships etc.), or
- Use of an existing system or system components to demonstrate some functions that will be included in the developed system.

Prototyping is comprised of the following steps:

1. Requirements Definition/Collection. Similar to the Conceptualization phase of the Waterfall Model, but not as comprehensive. The information collected is usually limited to a subset of the complete system requirements.
2. Design. Once the initial layer of requirements information is collected, or new information is gathered, it is rapidly integrated into a new or existing design so that it may be folded into the prototype.
3. Prototype Creation/Modification. The information from the design is rapidly rolled into a prototype. This may mean the creation/modification of paper information, new coding, or modifications to existing coding.
4. Assessment. The prototype is presented to the customer for review. Comments and suggestions are collected from the customer.

5. Prototype Refinement. Information collected from the customer is digested and the prototype is refined. The developer revises the prototype to make it more effective and efficient.
6. System Implementation. In most cases, the system is rewritten once requirements are understood. Sometimes, the Iterative process eventually produces a working system that can be the cornerstone for the fully functional system.

Criticisms of the Prototyping Model generally fall into the following categories:

1. Prototyping can lead to false expectations. Prototyping often creates a situation where the customer mistakenly believes that the system is "finished" when in fact it is not. More specifically, when using the Prototyping Model, the pre-implementation versions of a system are really nothing more than one-dimensional structures. The necessary, behind-the-scenes work such as database normalization, documentation, testing, and reviews for efficiency have not been done. Thus the necessary underpinnings for the system are not in place.
2. Prototyping can lead to poorly designed systems. Because the primary goal of Prototyping is rapid development, the design of the system can sometimes suffer because the system is built in a series of "layers" without a global consideration of the integration of all other components. While initial software development is often built to be a "throwaway" attempting to retroactively produce a solid system design can sometimes be problematic.

3.3 System goals analysis

To help us better present our system goals, we have employed the use a top-down hierarchy view of the project functionalities. We extract of goals from the given block diagram below.

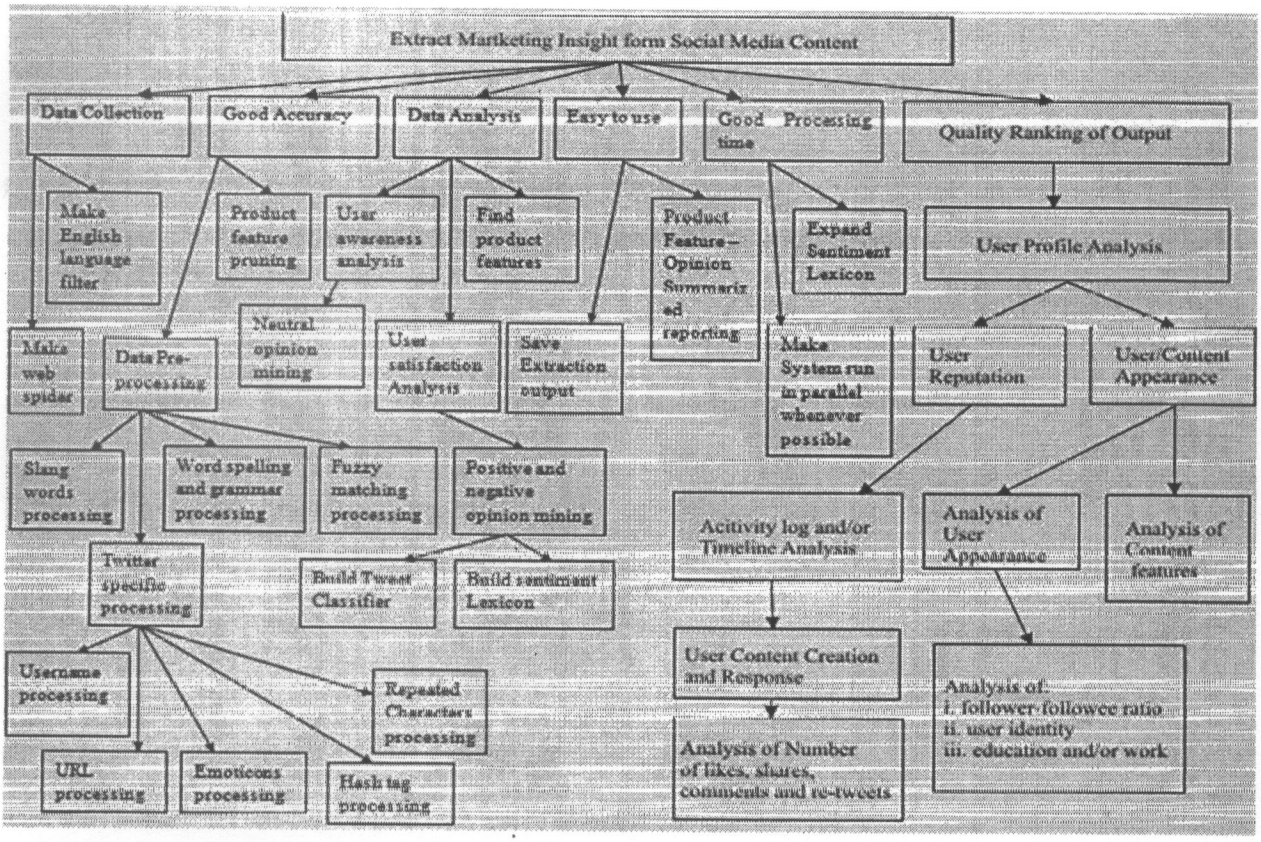

Figure 3.1: Goal Analysis Model

From the figure above (Fig 3.1), the block diagram illustrations the goal analysis model of this project. High-level goals are broken down to sub-levels as you go down the hierarchy. Given the trend, you realize how high-level goals are achieved. Some goals are functional (.e.g. Make Web Spider), while some are non-functional (e.g. Good Accuracy). It is from the goals presented above, that we elicit our system requirements from.

3.4 System functionalities

A functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. [28] Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation such as performance requirements, security, or reliability. Generally, functional requirements are expressed in the form "system must do <requirement>", while non-functional requirements are "system shall be <requirement>".

3.4.1 System functional requirements analysis

Section 3.3 of this chapter shows a goal analysis model; from there we can extract Functional requirements. This section shows the project requirements in tabular format. In each table, requirements, their description and the goals they are derived from are shown.

Goal Name:	Make web spider
Requirements:	1. The system must establish a data channel with Twitter Streaming API 2. The system must listen and download raw data(tweets) from the data channel for a specific product 3. The system must save the data
Explanation:	Twitter provides a streaming API. To download the data from twitter, the system must create a data channel then Listen for the data from the channel. When the data is available in the channel, the system must download the data and save it.

Table 3-1 web spider

Goal Name:	Make English language filter
Requirements:	1. The system must detect the language a tweet is written in 2. The system must save tweets written in English 3. The system must discard tweets not written in English
Explanation:	Twitter streaming API gives data written in many languages. This is because Twitter has users from across the globe. In this project, the system must detect the language of the tweets and drop messages written in other languages except in English.

Table 3-2 language filtering

Goal Name:	Slang words processing
Requirements:	1. Manually build a slang dictionary 2. The system must identify slang words in tweets 3. The system must replace slang words with their actual meaning.
Explanation:	Social media users use a lot of slang when writing their messages. Slang is the use of informal words and expressions that are not considered standard in the speaker's language or dialect but are considered acceptable in certain social settings. Example 'b4' means 'before'. The system must identify slang words using a slang dictionary and replace them with their meanings.

Table 3-3 slang words

Goal Name:	Word spelling and grammar processing
Requirements:	1. Manually build an English Dictionary 2. The system must identify wrongly spelled words 3. The system must replace wrongly spelled words with correct ones 4. The system must identify sentences with grammatical errors 5. The system must correct the grammar in sentences.
Explanation:	Tweets may be informal. Consequently, they contain a lot of spelling and grammatical errors. The system must identify wrong spellings using the English dictionary and replace them with correct words. The system must also identify wrong grammar in the sentences and corrects it.

Table 3-4 spelling and grammar errors

Goal Name:	Fuzzy matching processing
Requirements:	1. The system must identify fuzzy words in tweets 2. The system must identify the most popular fuzzy words 3. The system must replace all fuzzy words with popular fuzzy words
Explanation:	Fuzzy matching is used to deal with word variants or misspellings. For example, "jailbreak" and "jail-break" actually refer to the same word. The system must identify fuzzy words in tweets and replace them with the most popular fuzzy words.

Table 3-5 fuzzy matching

Goal Name:	Username processing
Requirements:	1. The system must identify words that starts with @ symbol 2. The system must delete the words that starts with @ symbol 3. The system must replace words that starts with @ symbol with a class token USERNAME

Explanation:	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. @Mukuka). The system must identify words that starts with the "@" symbol. For requirement 2, the system must delete words that start with the "@" Symbol. This is useful in product feature extraction. For requirement 3, the system must replace all words that start with the "@" symbol with an equivalence class token USERNAME. This is useful when doing tweet sentiment classification.

Table 3-6 Twitter usernames

Goal Name:	URL processing
Requirements:	1. The System must identify URLs in tweets 2. The System must delete URLs in tweets 3. The System must replace URLs in tweets with a class token URL
Explanation:	Twitter users very often include links in their tweets. The system must identify URLs in tweets. For requirement 2, the system must delete URLS. This is useful in product feature extraction. For requirement 3, the system must replace URLs with an equivalence class token URL. This is useful when doing tweet sentiment classification.

Table 3-7 Twitter URLs

Goal Name:	Emoticons processing
Requirements:	1. The system must identify negative emoticons in tweets 2. The system must identify positive emoticons in tweets 3. The system must delete both negative and positive emoticons in tweets 4. The system must replace negative emoticons with a class token SMILESAD

	5. The system must replace positive emoticons with a class token SMILEHAPPY
Explanation:	Twitter users very often use emoticons to express sentiments. An emoticon is a met-communicative pictorial representation of a facial expression which in the absence of the prosody serves to draw a receiver's attention to the tenor or temper of a sender's nominal verbal communication. E.g. □. The system must identify emoticons in tweets For requirement 3, the system must delete emoticons. This is useful in product feature extraction. For requirement 4 and 5, the system must replace emoticons with equivalence class tokens SMILESAD and SMILEHAPPY respectively. This is useful when doing tweet sentiment classification.

Table 3-8 Twitter emoticons

Goal Name:	Hash tag processing
Requirements:	1. The system must identify words that starts with # symbol 2. The system must delete words that starts with # symbol
Explanation:	Twitter users very often use the # Symbol to categorize their messages. The system must identify words that start with a # Symbol and delete them from tweets.

Table 3-9 Twitter hash tag

Goal Name:	Repeated Characters processing
Requirements:	1. The system must identify words that has character repetition 2. The system must correct the number of characters in the words 3. The system must replace any letter occurring more than two times in a row with two occurrences

Explanation:	Tweets contain very casual language. For example, you can write "hungry" with an arbitrary number of u's in the middle (e.g. Sooo, soorry). The system must identify character repetitions. For requirement 2, the system must correct these repetitions. This is useful in product feature extraction. For requirement 3, the system must replace any letter occurring more than two times in a row with two occurrences. This is useful when doing tweet sentiment classification.

Table 3-10 repeated characters

Goal Name:	Find product features
Requirements:	1. The system must POS tag nouns in tweets 2. The system must extract frequent product features 3. The system must extract infrequent product features
Explanation:	Part of the aim of this project is to extract product features in tweets. For requirement 1, the system must identify nouns in messages. In tweets, product features are mostly written as nouns. For requirement 2, the system must extract mostly talked about product features in a set of tweets. For requirement 3, the system must extract product features that are less talked about in a set of tweets.

Table 3-11 product feature extraction

Goal Name:	Product feature pruning
Requirements:	1. The system must identify irrelevant product features 2. The system must discard irrelevant product features
Explanation:	Product feature extraction can produce features which are irrelevant. The system must identify these irrelevant features and discard them.

Table 3-12 product feature pruning

Goal Name:	Neutral opinion mining
Requirements:	1. The system must find neutral opinions
Explanation:	Part of the aim of this project is to discover neutral opinions people express on product features. The system must find neutral opinions expressed on a product features by users.

Table 3-13 neutral opinions

Goal Name:	Positive and negative opinion mining
Requirements:	1. The system must POS tag adjectives in tweets 2. The system must find positive and negative opinions
Explanation:	Part of the aim of this project is to discover positive and negative opinions people express on product features. For requirement 1, the system must identify adjectives in messages. In tweets, positive and negative opinions are mostly expressed in adjectives. For requirement 2, The system must find positive and negative opinions expressed on a product features by users.

Table 3-14 positive and negative opinions

Goal Name:	Build Tweet Classifier
Requirements:	1. The system must download 20,000 twitter messages that contain positive emoticons 2. The system must download 20,000 twitter messages that contain negative emoticons 3. The system must download twitter messages. Manually annotate 3,000 positive messages 4. The system must download twitter messages. Manually annotate 3,000 negative messages

	5. The system must download twitter messages. Manually annotate 3,000 neutral messages 6. The system must train a classifier that can classify a tweets message into two classes; positive and negative 7. The system must train a classifier that can classify a tweets into three classes; positive, negative and neutral
	8. The system must save the trained classifiers
Explanation:	A Tweet Classifier is a trained algorithm that, given a twitter message, it can judge the sentiment polarity. You need to train the classifier on some data which already has been classified. Using this data, the classifier will learn how to classify unclassified data. In this project, 20,000 tweets with positive emoticons downloaded in requirement 1 must be used as positive data. 20,000 tweets with negative emoticons downloaded in requirement 2 must be used as negative data. For requirement 6, the system must train a classifier that can classify tweets into two classes; positive and negative using data downloaded in requirement 1 and data downloaded in requirement 2. For requirement 7, the system must train a classifier that can classify tweets into three classes; positive, negative and neutral using data downloaded in requirement 3, requirement 4 and requirement 5. For requirement 8, the system must save the trained classifiers.

Table 3-15 Tweet classifier

Goal Name:	Expand Sentiment Lexicon
Requirements:	1. The system must discover new opinion words not found in the Lexicon from tweets 2. The system must add the new opinion words to the Lexicon
Explanation:	The Lexicon does not contain all the words tweets use to express their opinions. Hence the system must be able discover new opinion words from tweets and add them to the lexicon.

Table 3-17 expand a lexicon

Goal Name:	Product Feature – Opinion Summarized reporting
Requirements:	1. For each product feature extracted, the system must count the number of tweets that have expressed positive, negative and neutral opinions. 2. The system must associate the product features to the tweets messages that contains them.
Explanation:	For requirement 1, the system must count the number of tweets that have expressed positive, negative and neutral opinions about product features. For requirement 2, The system must associate extracted product features to tweets that contain them. This is important so that the users can view the messages to see why existing customers like it and/or what they complain about.

Table 3-18 product feature – opinion summary report

Goal Name:	Rank output by feature statistics
Requirements:	1. The system must count how many times each product feature appears in the set of tweets 2. The system must rank frequent product features higher than infrequent product features 3. The system must rank a product feature appearing more in tweets higher than a product feature appearing less
Explanation:	The system must make a product feature with a higher rank appear first than a product feature with a lower rank.

Table 3-19 feature ranking

Goal Name:	Rank output by user statistics
Requirements:	1. The system must count how many followers a Twitter user has 2. The system must count how many followees a Twitter user has 3. The system must calculate the follower-followee ratio 4. The system must rank a tweet with a higher follower-followee ratio higher than a tweet with a lower ratio
Explanation:	Twitter has the following-follower model which allows one to follow and receive new messages from others. This makes information to spread fast. The system must count how many follows a user and how many a user followers and then calculate the follower-followee ratio. A message coming from a user with a higher ratio, the system must make it appear first than the message coming from a user with a lower ratio.

Table 3-20 user ranking

Goal Name:	Rank output by message statistics
Requirements:	1. The system must count how many times each tweet is reposted 2. The system must rank a tweet with a higher repost count higher than a tweet with a repost count
Explanation:	Twitter allows message to be reposted which makes message move faster. When a user receives a message from another user, he/she can repost it to his followers. The system must count how many times a message has been reposted. A message with a higher repost count, the system must make is appear first than a message with a lower count.

Table 3-21 message ranking

Goal Name:	Save Extraction output

Requirements:	1. The system must save the extracted product features and their opinion information
Explanation:	After the Product Feature – Opinion Summarized report is generated; the system must save it so that a user can open it later without running the extraction again.

Table 3-22 save output

3.4.2 System non-functional requirements analysis

Non-Functional Requirements are derived from the goals given in section 3.1 of this chapter. This section shows the project requirements in tabular format. In each table, requirements, their description and the goals they are derived from are shown.

Goal Name:	Good Accuracy
Requirements:	1. The system must give 80% accuracy on product feature extraction 2. The system must give 80% accuracy on opinion mining 3. The system must give 80% accuracy on tweet classification
Explanation:	The accuracy of the system on product feature extraction, opinion mining and tweet classification must be eighty percent or better.

Table 3-23 accuracy

Goal Name:	Easy to use
Requirements:	1. The user must be able to run the product features extraction in less than 3 mouse clicks 2. The system must provide defaults for system settings 3. The system must display product features, their opinion information and tweet that contain them simultaneously to make comparisons easier

	4. System must maintain history of the product feature extraction for later reference 5. User must be able to customize the graphical user interface 6. The system must provide a user help manual
Explanation:	The system must be easy to use.

Table 3-24 easy to use

Goal Name:	Good Processing time
Requirements:	1. The system must produce the Product Feature – Opinion Summarized report from a set of 8,000 tweets in less than 30 minutes running on the system with RAM of 2GB, and Processor of Intel ® Core ™ i3-2310M CPU @ 2.10GHz
Explanation:	A Product Feature – Opinion Summarized report is a report which shows extracted product features and their opinion information together with the tweets that contains them.

Table 3-25 processing time

Goal Name:	Make System run in parallel whenever possible
Requirements:	1. System must download tweets from twitter in parallel 2. The system must extract product frequent and product infrequent feature in parallel 3. The system must classify tweets in parallel 4. The system must run opinion mining on product frequent and product infrequent feature in parallel

Explanation:	For requirement 1, the system must download tweets from twitter in parallel For requirement 2, the system must extract frequent product features and infrequent product features in parallel For requirement 3, the system must classify tweets in parallel For requirement 4, the system must discover opinions on frequent product features and infrequent product features in parallel

Table 3-26 parallelism

Goal Name:	User Login/Logout
Requirements:	1. The system must allow a remote user to create an account 2. The system must allow a remote user to log into he/her account 3. The system must allow a remote user to safely logout without losing data
Explanation:	Remote users should be able to create accounts by signing up and be able to log in. when done, the system should further all them to logout while it safe saves there datasets and results.

Table 3-27 User login

3.4 Development tools and software

In this section, we present the software development tools and libraries that were used in the project. The main language of implementation used is java and to extend the system applications hosted on the server we used Servlets. In order to present a dynamic and interactive interface on

the web, the Java Server Pages (JSP) technology is employed. The following are some of the many software tools and libraries used.

1. NetBeans IDE: An integrated development environment or interactive development environment is a software application that provides comprehensive facilities to computer programmers for software development.
2. (POS Tagger) : A Part-Of-Speech Tagger is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc.
3. Streaming API: Provides streams of the public data flowing through Twitter for data mining.
4. MySql:
5. Language tool: is open Source proofreading software for English, French, German, Polish, and more than 20 other languages.
6. Libsvm: open source machine learning library used for classification.
7. Stanford parse: this is a natural language parser library that works out the grammatical structure of sentences.
8. Lingpipe: this is tool kit for processing text using computational linguistics; it can identify place, location or people.
9. Commons-lang: a library that provides extra helper utilizes for the java.lang API
10. Jsoup: this is a Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and jquery-like methods.

4.0 SYSTEM DESIGN

4.1 Explanation of the proposed

In this section, we explain the proposed system by using a context diagram. It defines the boundary between the system, or part of a system, and its environment, showing the entities that interact with it. This diagram is a high level view of a system.

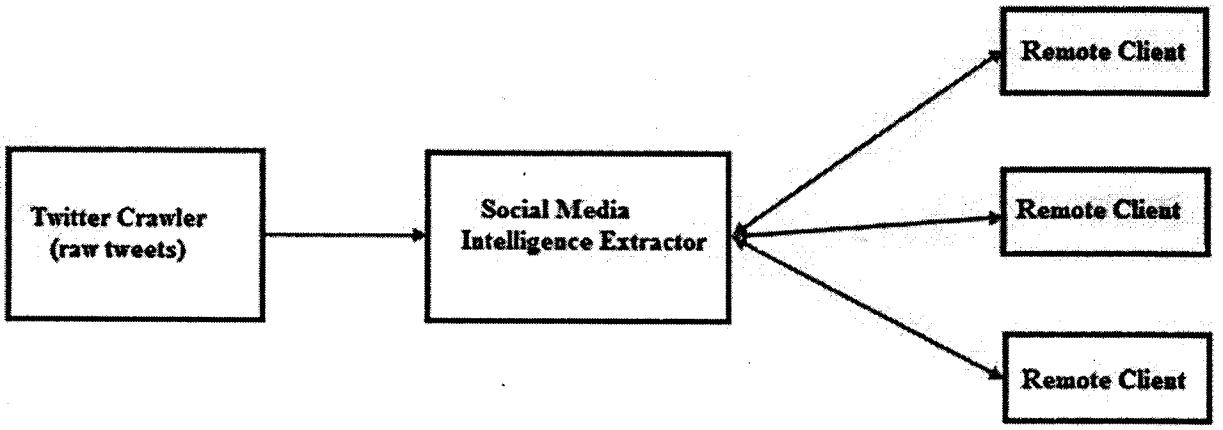

Figure 4.1: Context Model

The crawler in the above diagram provides raw (unprocessed) tweets; this is achieved by using the web API that Twitter provides. Remote clients request for their datasets of interest and receive JSON objects; these are then presented to the user in a well formatted manner by the client side intelligence.

4.2 Software level Architectural Design

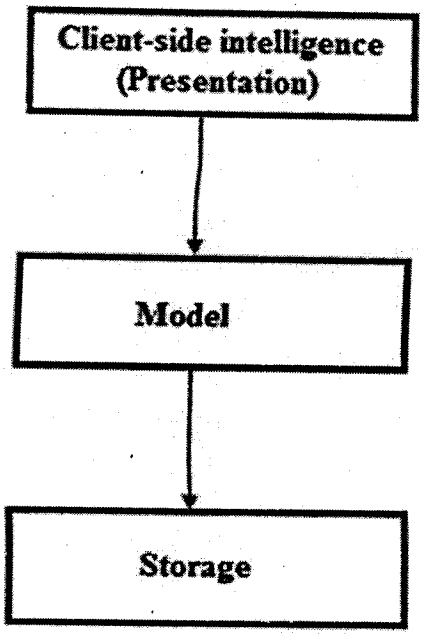

Figure 4.2: Large-scale System Architecture

The system is separated into three distinctive layers. The top layer is the presentation layer containing client intelligence. It is responsible for handling user – system interactions. This layer allows the user to download data from Twitter by specifying the product name. The presentation layer is also responsible for invoking the whole system operational procedures. When the system extracts product features and user opinion information, it responds to the presentation layer by sending a JSON object contain all the processed information. The presentation layer is then for formatting and displaying of the results to the remote user. It goes without mention that the presentation layer heavily relies on the Model layer as shown in the diagram above. The Model layer implements social media extraction logic. It is responsible for downloading the data from Twitter and processing it into Standard English. It further extracts product features from the data together with user opinions. It saves the data and the extraction output in the Storage layer.

4.3 Modular design of the System functions

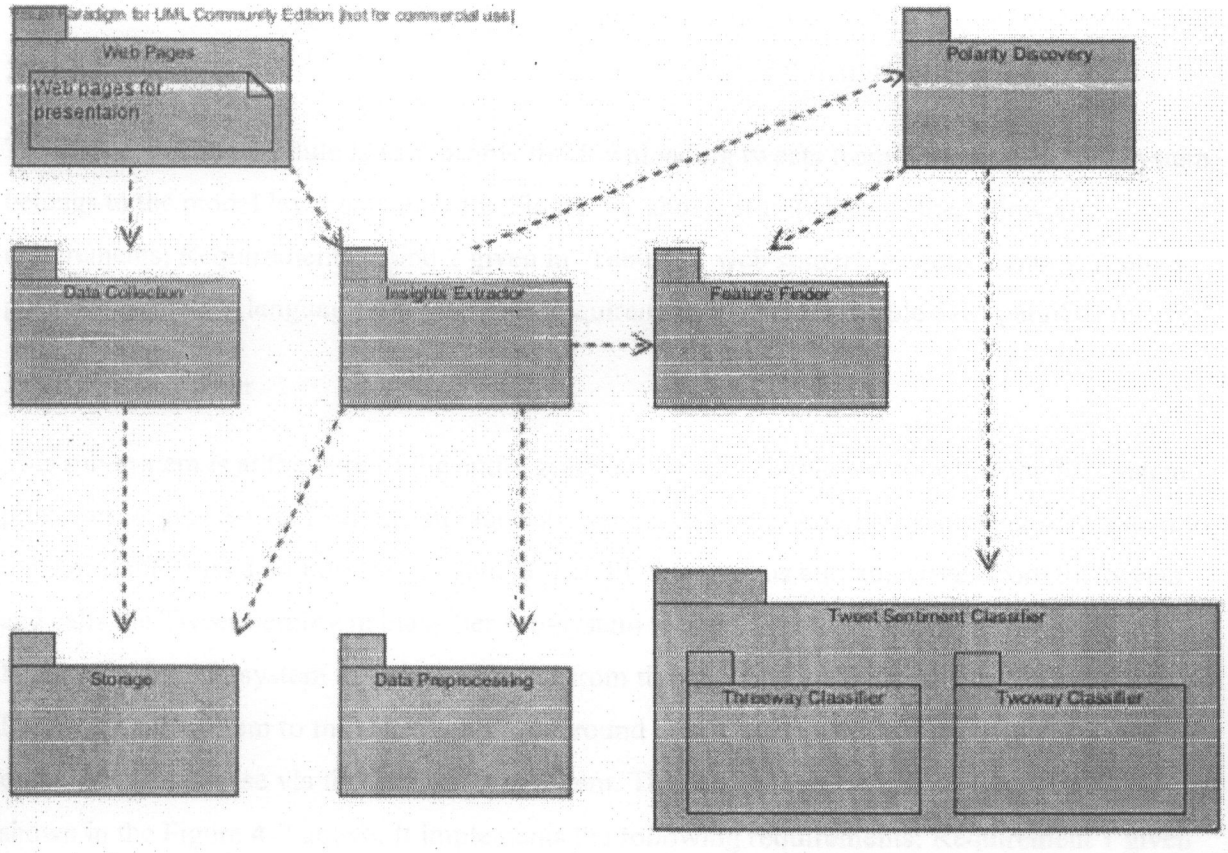

Figure 4.3: System Package Diagram

The system has the following sub-system;

1) Presentation (Web pages)

The presentation module is responsible for handling user-system interactions. It provides a GUI interface for the remote user to interact with the system. It allows the user to download the Twitter data, change system settings and extract product features and user opinions from the downloaded tweets. It is also responsible for displaying the system output. This sub-system belongs to the presentation layer shown in the Figure 4.2 above. It implements the following

requirements; Requirements1, 2 and 3 given in 'Table 3-19 feature ranking'. Requirements 1, 2, 3, and 4 given in 'Table 3-20 user ranking'. Requirements 1 and 2 given in 'Table 3-21 message ranking' and Requirements1, 2, 3, 5 and 6 given in 'Table 3-24 easy to use'.

2) Data Collection

The data collection module is responsible for downloading tweets from Twitter. This sub-system belongs to the model layer shown in the Figure 4-2 above. It implements the following requirements; Requirements 1, and 2 given in 'Table 3-1 web spider'. Requirements 1, 2 and 3 given in 'Table 3-2 language filtering' and Requirement 1 given in 'Table 3-26 parallelism'.

3) Insights Extractor

This sub-system is at the core of the entire system. It is responsible for coordinating the system processes. It gets tweets from the storage sub-system, calls the Data Preprocessing sub-system to transform informal data into formal data. It then POS tag nouns and adjectives from the tweets and calls the Tweet Sentiment classifier sub-system to start classifying them. It then calls the Feature Finder sub-system to extract features from the tweets. Finally it calls the Polarity Discovery sub-system to mine user opinions around the extracted product features and saves the output in the database via the Storage sub-system. This sub-system belongs to the model layer shown in the Figure 4-2 above. It implements the following requirements; Requirement 1 given in 'Table 3-11 product feature extraction', Requirement 1 given in 'Table 3-14 positive and negative opinions', Requirement 4 given in 'Table 3-24 easy to use'. Requirement 1 given in 'Table 3-25 processing time' and Requirement 3 given in 'Table 3-26 parallelism'.

4) Data Preprocessing

Data Preprocessing is responsible for transforming informal tweets into formal ones. It corrects grammatical and spelling mistakes in tweets. It also removes meaningless words e.g. web URLS. This sub-system belongs to the model layer shown in the Figure 4-2 above. It implements the following requirements; Requirements 2 and 3 given in 'Table 3-3 slang words'. Requirements 2, 3, 4 and 5 given in 'Table 3-4 spelling and grammar error'. Requirements 1 and 2 given in 'Table 3-6 Twitter usernames'. Requirements 1 and 2 given in 'Table 3-7 Twitter URLs'.

Requirements 1, 2 and 3 given in 'Table 3-8 Twitter emoticons'. Requirements 1 and 2 given in 'Table 3-9 Twitter hash tag' and Requirements 1 and 2 given in 'Table 3-10 repeated characters'.

5) Feature Finder

This sub-system is responsible for extracting product features from twitter messages (tweets). It also does feature pruning to get rid of irrelevant features. It employs two techniques to extract features; Team frequency technique and Meronym patterns. This sub-system belongs to the model layer shown in the Figure 4-2 above. It implements the following requirements; Requirement 1, 2 and 3 given in 'Table 3-5 fuzzy matching'. Requirements 2 and 3 given in 'Table 3-11 product feature extraction'. Requirements 1 and 2 given in 'Table 3-12 product feature pruning'. Requirement 1 given in 'Table 3-23 accuracy' and requirement 2 given in 'Table 3-26 parallelism'.

6) Tweet Sentiment Classifier

This module is responsible for classifying an entire tweet into positive, negative and neutral classes. It trains on some given data and saves the trained classifier. This sub-system belongs to the model layer shown in the Figure 4-2 above. It implements the following requirements; Requirements 1 and 3 given in 'Table 3-6 Twitter username'. Requirements 1 and 3 given in 'Table 3-7 Twitter URLs'. Requirements 1, 2, 3 and 5 given in 'Table 3-8 Twitter emoticons'. Requirements 1 and 2 given in 'Table 3-9 Twitter hash tag'. Requirements 1 and 3 given in 'Table 3-10 repeated characters'. Requirements 1, 2, 3, 4, 5, 6, 7 and 8 given in 'Table 3-15 Tweet classifier' and Requirement 3 given in 'Table 3-23 accuracy'.

7) Opinion Discovery

This module is responsible for customer opinion mining. It discovers opinions expressed by users on product features. It is also responsible for generating the Product feature – opinion summarized report. This report is the system output which gives for each product, the number of positive, negative and neutral tweets. This sub-system belongs to the model layer shown in the Figure 4.2 above. It implements the following requirements; Requirement 1 given in 'Table 3-13 neutral opinions', Requirement 2 given in 'Table 3-14 positive and negative opinion'. Requirements 1 and 2 given in 'Table 3-17 expand a lexicon'. Requirements 1 and 2 given in

'Table 3-18 product feature – opinion summary report'. Requirement 2 given in 'Table 3-23 accuracy' and requirement 4 given in 'Table 3-26 parallelism'.

8) Storage

Storage is responsible for storing downloaded tweets and the system output (Product feature – opinion summarized report). This sub-system belongs to the Storage layer shown in the Figure 4-2 above. It implements the following requirements; requirement 3 given in 'Table 3-1 web spider' and requirement 1 given in 'Table 3-22 save output'.

4.4 System structure Data Flow Diagrams and Entity Relation Diagram

Under this segment, we present the database design using entity relation models and the actual appearance of the populated database tables.

4.4.1 Entity Relation Diagram (ERD)

The information about this dataset is stored in the dataset table. processedtweets table holds the tweets that have been processed by the system through the Data Preprocessing sub-system given in Figure 4-3. feature table holds the product features extracted by the system. Figure 4-6 below shows the important entities, relationships, and attributes in the domain. rawtweets table holds the tweets dataset that is downloaded from Twitter Every time the system runs, it maintains the history of the extraction. The history data is kept in the history table. The relationship between the extracted features, user opinion and the tweets is kept in tweetswithfeatures table. The relationship between the features and tweets is a many to many relationship. A user may have multiple datasets to his name as shown in the diagram below.

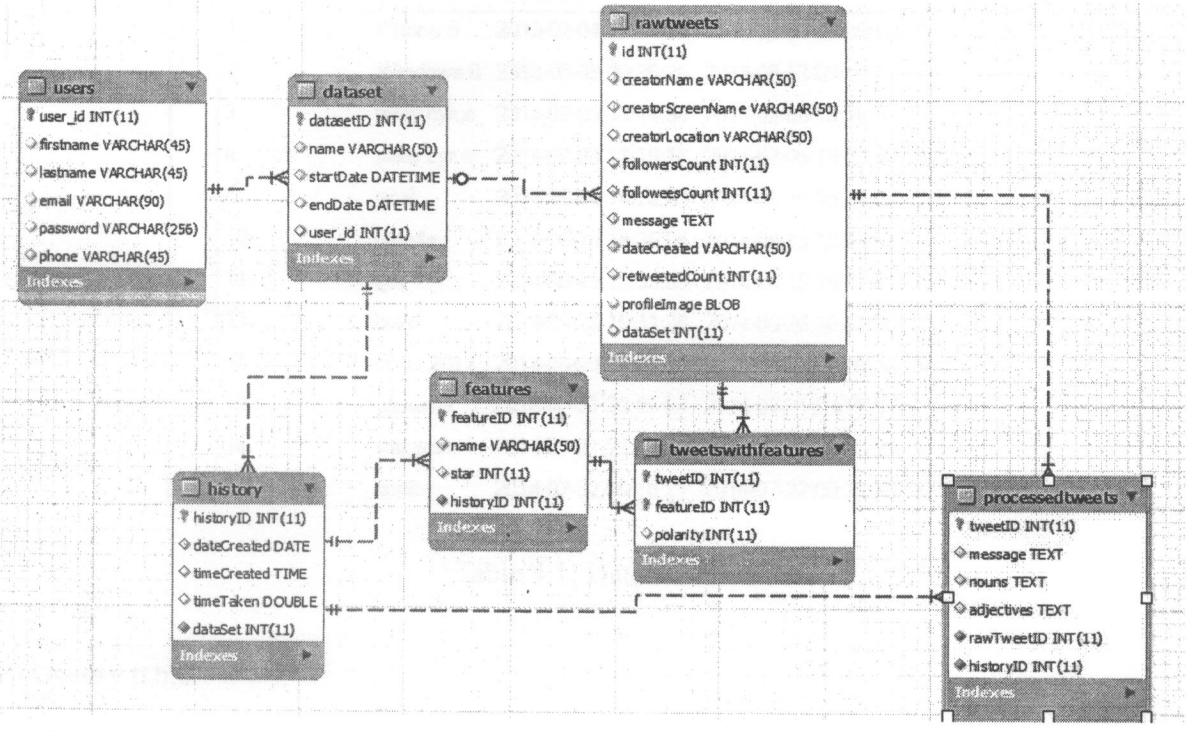

Figure 4.4: Entity relation diagram

4.4.2 Physical Model Design

The physical database design shows how the entity relation model (conceptual model) given above is physically implemented as base relations/tables in the MySQL DBMS. InnoDB was chosen as the storage engine because it provides the standard ACID compliant transaction features, along with foreign key support which is important for specifying constraints on the data. Because Twitter messages can be written in many languages, UTF8 Char set was chosen for holding textual data.

1) Dataset table

In the table below, we present dataset table. It contains twelve rows of sample data.

	datasetID	name	startDate	endDate	user_id
▶	1	iPhone 5	2013-02-04 07:07:28	2013-02-08 18:00:10	1
	2	Windows 8	2013-05-09 00:30:26	2013-05-13 09:40:11	2
	3	miley cyrus	2014-02-05 13:14:58	2014-02-05 13:18:58	3
	4	miley cyrus	2014-02-05 13:19:16	2014-02-05 13:23:44	3
	8	goal	2014-06-16 14:26:09	2014-06-16 15:28:09	1
	10	nigeria	2014-06-19 18:36:04	2014-06-19 18:36:58	1
	11	get	2014-06-19 19:39:59	2014-06-19 19:42:41	2
	13	brazil	2014-06-20 10:41:36	2014-06-20 10:42:30	2
	14	brazil 2014	2014-06-20 12:04:13	2014-06-20 12:09:36	3
	15	germany	2014-07-21 23:06:29	2014-07-21 23:18:07	1
	16	star wars	2014-07-21 23:29:55	2014-07-21 23:38:23	1
	23	apple	2014-07-22 00:18:21	2014-07-22 00:31:06	1

Table 4.1: Dataset Table

2) IIistory table

Given below is the history table. It contains nine rows of samplc data.

	historyID	dateCreated	timeCreated	timeTaken	dataSet
▶	1	2013-05-27	09.15.42	33.612716666666664	1
	2	2013-05-27	10:07:41	17.16505	2
	3	2014-07-24	00:28:47	2.6473333333333335	23
	4	2014-07-24	14:17:55	4.559866666666666	10
	5	2014 07-24	14:22:54	0.29118333333333335	10
	6	2014-07-24	15:03:51	13.1142	24
	7	2014-07-25	14:47:42	0.3806333333333333	25
	8	2014-08-03	22:10:58	0.994	39
	9	2014-08-04	15:39:32	14.887066666666666	35

Table 4.2: History Table

The table above has one constraint on dataSet field. This column is a foreign key to the datasetID field in dataset table given in "Table 4-1 dataset table". This saves to link the dataset and history tables.

3) Rawtweets table

id	creatorName	creatorScreenName	creatorLocation	followersCount	followeesCount	message
1	Tom Costello	TM_Costello	Dundee, Scotland	117	118	@hardgraft Finally received my iPhone 5 case! Thanl
2	Zeling	ZelingPotter		226	237	Need somebody with iPhone 5 charger in school to s
3	TechXpert	Digixp	in front of my mac	275	206	iPhone 5 Jailbreak Tool To Be Released Tomorrow h
4	Bird Information	BirdInformation		564	582	RT @iaindownie: A few people reporting @BirdTrack
5	Geeky Gadgets	GeekyGadgets		9411	267	iPhone 5 Jailbreak Coming Today http://t.co/EYw8S
6	Diode-D-Bot	TheGeeklyReport	on a spaceship near Saturn	843	1807	GEEKS-R-US! iPhone 5 Jailbreak Coming Today; It k
7	Neil Bent	OriginalDuffer	iPhone: 53.804055,-1.487538	21	0	Still waiting for iPhone 5 jailbreak.
8	John Morgan	IPhonesAppMgr	Baton Rouge, LA	625	1026	Dropped iPhone 5 Into Septic Tank Today http://t.c
9	lauren gale.	laurengale_	Thatcham	1463	1610	RT @AlexDopeCreed: Big up the back of the car IPt
10	Jill Carter Beauty	jcarterbeauty		49	36	This should be a hoot - me trying to swap my iPhone
11	Pakiads	pakiads		1065	0	Brand new unlocked Apple iPhone 5 64GB - Karachi

Table 4.3: Rawtweets Table

Given above is the rawtweets table. It contains eleven rows of sample data. The table above has one constraint on dataSet field. This column is a foreign key to the datasetID field in dataset table given in "Table 4.1 dataset table". This saves to link the rawtweets and dataset tables.

4) Features table

Below is a features table. It contains two rows of sample data.

featureID	name	star	historyID
1	case	3	1
2	Jailbreak	3	1
3	iOS	3	1
4	BLACK	3	1
5	iPad	3	1
6	Pack	3	1
7	Jailbreak iOS	3	1
8	juice Pack	3	1
9	Mophie juice Pack	3	1
10	White	3	1
11	juice Pack Helium	3	1
12	Crystal	3	1

Table 4.4: Features Table

The table above has one constraint on historyID field. This column is a foreign key to the historyID field in history table given in "Table 4-2 history table". This saves to link the features and history tables.

5) Processedtweets table

Given below is the processedtweets table. It contains ten rows of sample data.

	tweetID	message	nouns	adjectiv
▶	38125	Finally received my iPhone5 case! Thanks! Now I'Ve just got to use it as much as the last one.	case	last
	38126	Need somebody with iPhone5 charger in school to save the phone.	chargerschool	
	38127	IPhone5 Jailbreak Tool To Be Released Tomorrow	JailbreakTool	
	38128	A few people reporting application on iPhone5 crashing without OPEN. Can anyone reply to me directly if it is Wor.	applicationOPENreplyWor	few
	38129	IPhone5 Jailbreak Coming Today	Jailbreak	
	38130	GEEKSRUS! IPhone5 Jailbreak Coming Today: It looks like the iPhone5 Jailbreak will be released some time tod...	GEEKSRUSJailbreakJailbreaktod	
	38131	Still waiting for iPhone5 Jailbreak.	Jailbreak	
	38132	Dropped iPhone5 Into Septic Tank Today	Septic	
	38133	Big up the back of the car iPhone5 crew!	Bigbackcarcrew	
	38134	This should be a hoot me trying to swap my iPhone for to my new iPhone5.	hoot	new

Table 4.5: Processedtweets Table

The table above has one constraint on historyID field and one on the rawTweetID field(not shown due to space). These columns are foreign key to the historyID field in history table given in "Table 4-2 history table" and id field in the rawtweets table given in "Table 4-3 rawtweets table" respectively. These saves to link the processedtweets table to the history table and rawtweets table respectively.

6) Tweetswithfeatures table

Below is a tweetswithfeatures table. It contains thirteen rows of sample data.

	tweetID	featureID	polarity
▶	1	1	4
	2	26	-1
	2	105	-1
	3	2	-1
	3	107	-1
	3	109	-1
	3	125	-1
	4	115	0
	4	228	0
	5	2	-1
	6	2	-1
	6	107	-1
	6	109	-1

Table 4.6: Tweetswithfeatures Table

The table above has one constraint on featureID field and one on the tweetID field. These columns are foreign key to the featureID field in features table given in "Table 4-4 feature table" and id field in the rawtweets table given in "Table 4-3 rawtweets table" respectively. These saves to link the tweetswithfeatures table to the features table and rawtweets table respectively. It holds the many-to-many relationship that exists between the features and the tweet. A feature can belong to many tweets and one tweet can contain more than one feature.

7) Users table

Below is a user table. It contains two rows of sample data.

	user_id	firstname	lastname	email	password	phone
▶	1	John	Doe	jdoe@mail.com	pass	NULL
	2	Victor	Chulu	vchulu@gmail.com	pass	26097272201

Table 4.7: Users Table

The users in the table are uniquely identified using the user_id. For each user, we store the attributes shown in the above table.

4.5 User Interface Design

This section shows the user interfacc dcsign. This will bc thc interface that a remote user will user to interact with the system. We will use Cascading Style Sheet 3 (CSS3), AJAX and JAVASCRIPT to design the interface. 80% of the GUI will be in white and light grey strips. The progressive bars will be in green stripes and the over progressive bar in red.

The design of thc user interface is motivated by the system requirements. Requirement 3 in *'Table 3-24 easy to use'*, states that the system must display product features, their opinion information and tweets that contain them simultaneously to make comparisons easier. To fulfill this requirement, the system interface is logically alienated into sections that make it easy to understand as shown in Figure 4-8 below.

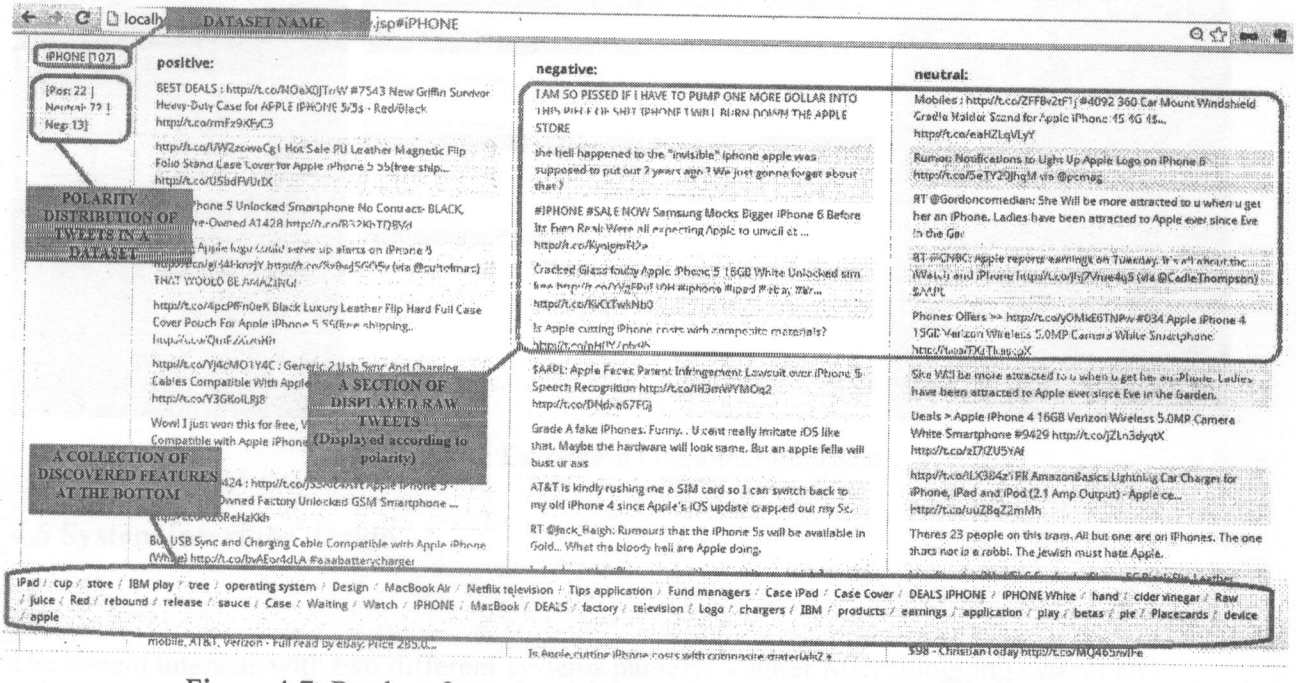

Figure 4.7: Product feature – opinion summarized repot screen shot

The product name appears in the top right corner. Just below the product or dataset name is the polarity distribution of the tweets in a given dataset. There are three states namely positive, negative and neutral. Against each of them is a number indicating the number of tweets in that category. At the bottom of the pages is a collection of features extracted from a dataset. Tweets are organized and displayed according to their polarity.

Requirement 1 in *'Table 3-24 easy to use'*, states that the user must be able to run the product features extraction in less than 3 mouse clicks. To fulfill this requirement, the extraction GUI window is divided into two columns. At the right end of the top bar, are clickable links (Home, History, Datasets, Sign Out). The system takes time to extract the products and process opinion mining; hence to keep the user informed concerning the progress, the system shows the progress to the user. The extraction alert window has multiple sub-process progress bars as shown below. The sub-progressive bars are in green and the overall bar is in red and at the bottom.

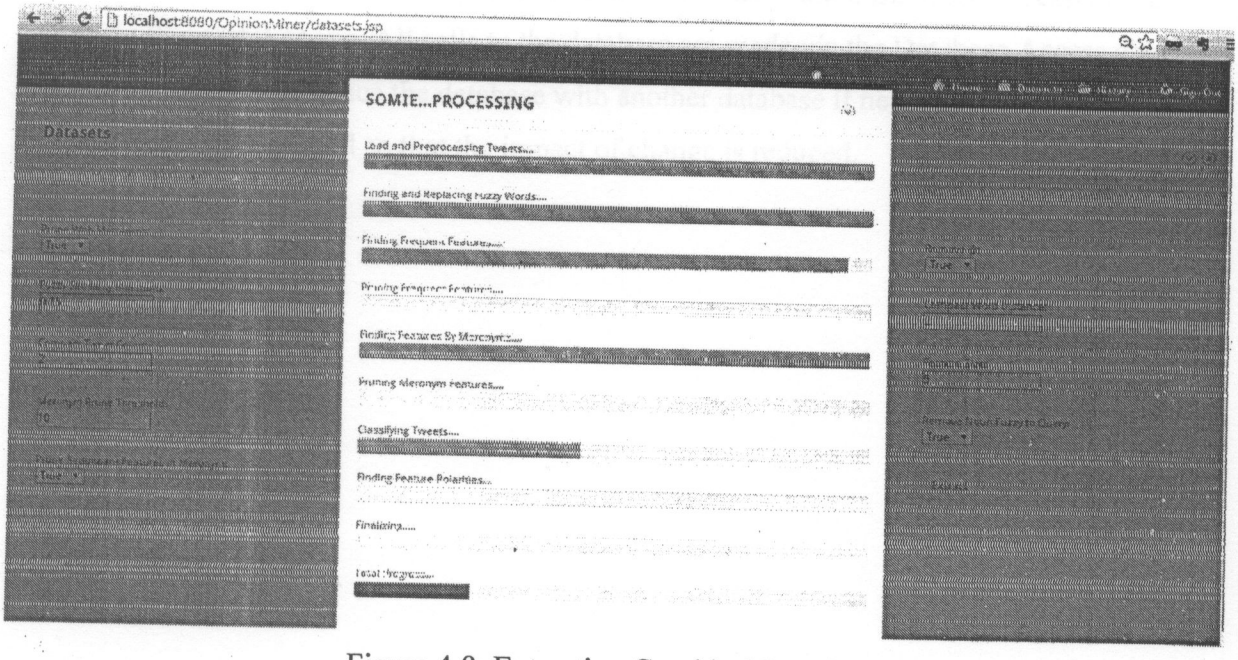

Figure 4.8: Extraction Graphical Interface

4.6 System interface design

The system interacts with two different systems namely; Twitter Microblogging system and MySQL database. The system boundary is defined such that the system is modular (the system acts more or less as an independent unit).

1) Twitter Microblogging system Interface

To interact with Twitter on the internet, the system makes web API calls that Twitter implements. To make the calls, the system uses only one class called TwitterStreamer class shown in Figure 4-4 above. This ensures that any changes made by Twitter will only affect one class in the system.

2) Database Interface

To interact with the database, the system uses only one class called DatabaseAccessObject shown in Figure 4-4 above. All calls to the database are made via the Database Access object. This makes it easy to replace the database with another database if need be. The interfaces to other systems are designed so that the impact of change is reduced.

5.0 SYSTEM IMPLEMENTATION

This chapter gives a detailed description of the development of the system and how each sub-system of a system is designed to perform its functions. It furthermore incorporates screenshots of a developed system

5.1 Technical details of the developed system and screen shots

5.1.1 Data Collection Module

Twitter implements a web streaming API that can be called to get its data. By using this, the data collection module is responsible for downloading data from Twitter. To achieve this, the application needs to register to Twitter by getting a Consumer key. Using this key, it can make calls on the API and receive the data. But the data received from the streaming API is in different languages. In this project, we are only interested in tweets written in English. To achieve this function, the Data collection module implements four classes shown in Figure 5-1 below.

1) TwitterStreamer Class

This class establishes a data channel with Twitter; it connects to Twitter web API and downloads tweets. To achieve this, this class constantly listens from the data channel and when the data is available in the channel, it downloads the Twitter message (tweet). This class also connects to the GUI of the system. As the data comes in, it displays every tweet on the screen including the image of the author. It also keeps a counter which notifies the user of how many tweets have been downloaded. The Figure below is a screen shot showing a live download session.

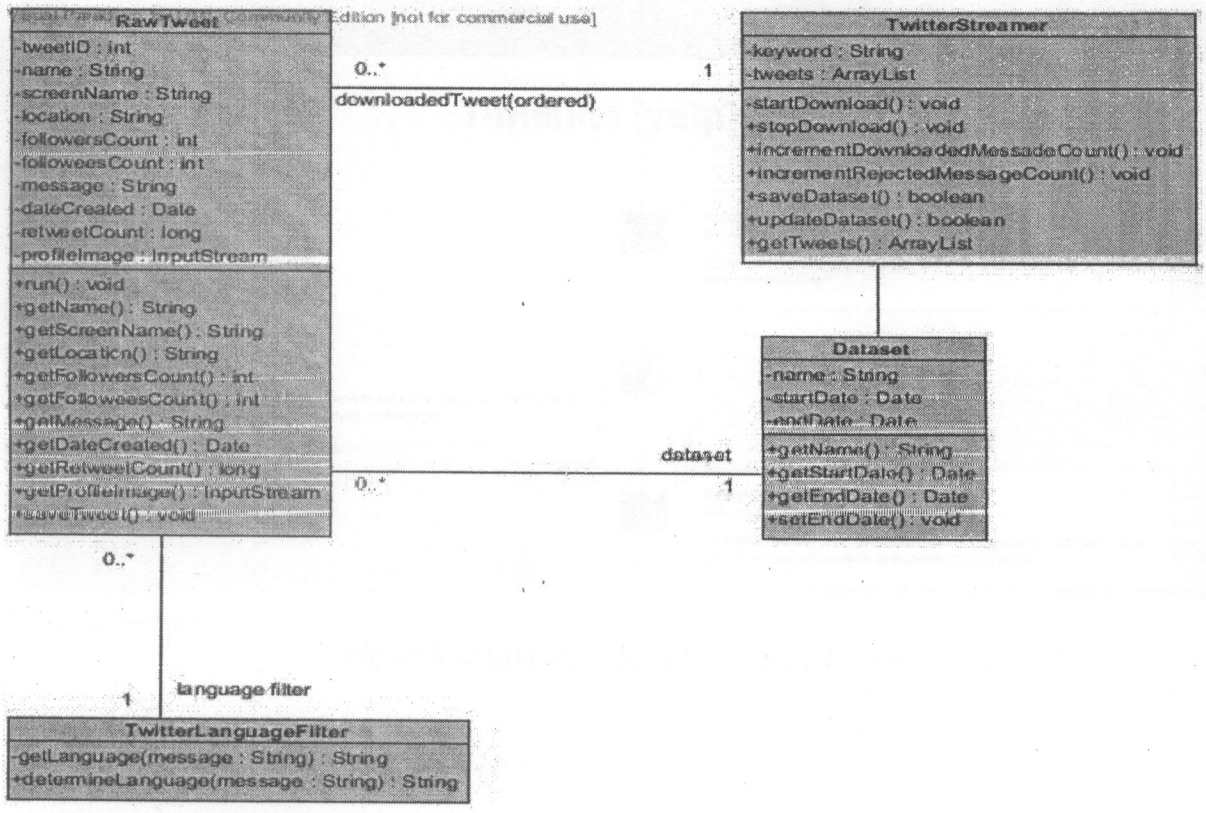

Figure 5.1: Data Collection Class Diagram

4) TwitterLanguageFilter Class

This class is called by the RawTweet class to detect the language in which a given message is written. Detecting the language of a Twitter message is not easy because a message can contain one word or many words. European languages use a similar alphabetical order with some small differences. So if a message contains only one word, it is difficult to detect its language.

To perform this function, this class uses the English dictionary to count how many English words and many non- English words are in a message. If English words equals or is more than non-English words, a message is accepted as written in English otherwise it is discarded. Because messages from Twitter are informal and contains a lot of spelling errors, this class first transforms an informal message into a formal one before detecting the language. The transformation includes correcting the spellings, correcting the grammar, removing URLs, removing emoticons and removing usernames. This technique used proved highly effected in detecting the language.

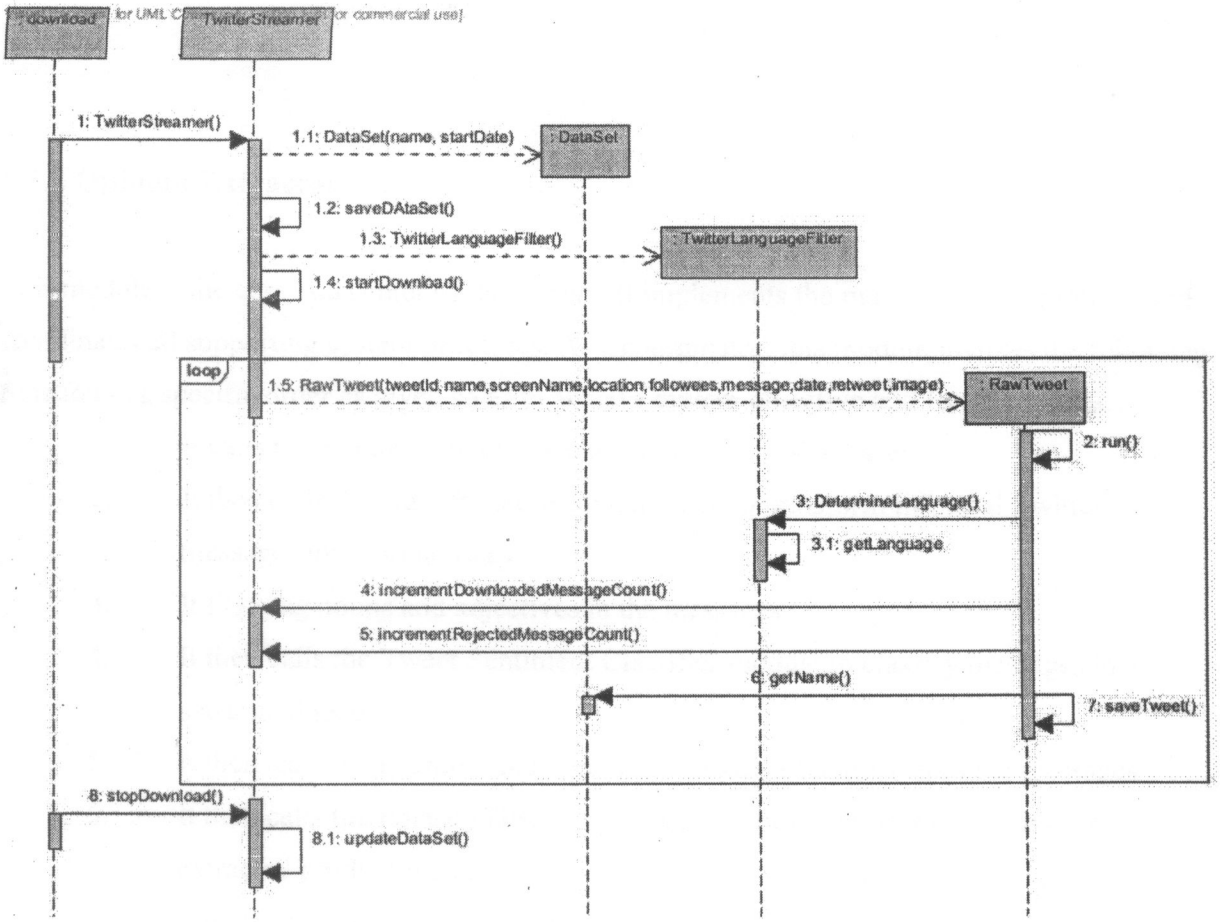

Figure 5.3: Data collection sequence diagram

The Figure above shows the sequence diagram of the Data collection module. When data is downloaded, it is stored to the database as a dataset. This is accomplished by the storage module given later in this chapter. A request comes from the remote client in the Presentation layer invoking the TwitterStreamer class to start the data download. TwitterStreamer creates the Dataset object using the name of the data to download and saves it in the database. It then creates the TwitterLanguageFilter class and passes it to the RawTweet object. The TwitterStreamer object will be in a loop downloading the data until the user stops the download. The download is stopped in the TwitterStreamer via the stopDownload method.

5.1.2 Opinion Extractor

This module is the core and center of the system. It implements the main system algorithms and coordinates all supporting system processes. As a coordinator, this module invokes the following function in a specific order as given so as to achieve system objective and functionalities.

1. It calls the Storage module to load data from the database
2. It then calls the Data Preprocessing module to transform informal Twitter messages into formal ones
3. It POS tag nouns and adjectives in the messages
4. It then calls the Tweet Sentiment Classifier module to classify messages in a separate thread
5. It then calls the Feature Finder module to extract features from the messages
6. It then calls the Polarity Discovery module to discover user opinions of the extracted product features
7. Finally, it calls the Storage module again to store the system output.

To achieve the above itemized functions, this module implements the following classes as shown in the class diagram Figure 5.4 given on the next page;

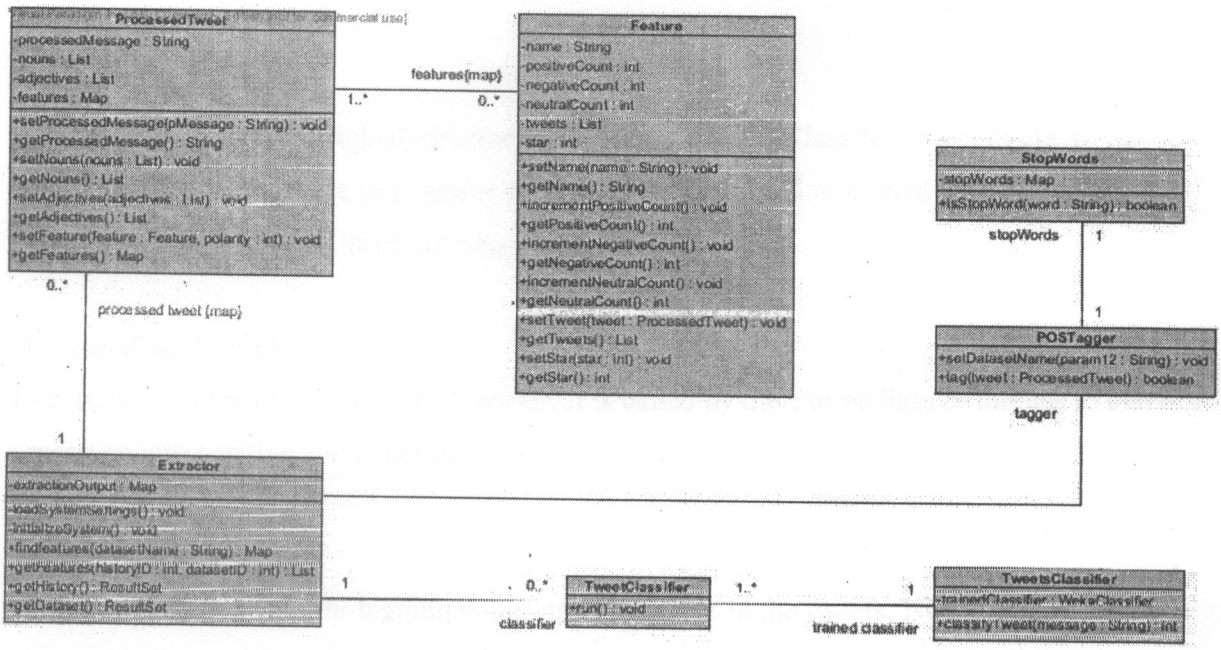

Figure 5.4: Opinion Extractor Class Diagram

1) Extractor (Opinion Extractor) class

It is responsible for coordinating system processes. It is also responsible for handling communication between the system model subsystem and the Presentation layer.

2) ProcessedTweet class

The system transforms an informal Twitter message into a formal one by correcting spelling and grammar and other things. The system also POS tag nouns and adjectives. All this information is stored in this class.

3) Feature class

When the system extracts features from the messages, each feature is stored as a Feature object. Every Feature object contains the feature name, how many messages are negative about this Feature, how many messages are positive about the Feature and how many messages are neutral. It also contains a list of all the messages that contains this Feature. A Feature class has a many-to-many relationship with ProcessedTweet class. A Feature can belong to many messages and a message can contain more than one features.

4) TweetClassifier class

For every message, the InsightsExtractor class creates a TweetClassfier object to classify a message into either positive or negative classes. This classification is used in the Polarity Discovery process. TweetClassifier runs as a thread.

5) TweetsClassifier class

TweetsClassifier holds the trained classifier. It is called by the TweetClassier threads to classify a message into positive and negative classes.

6) POSTagger class

This class is responsible for tagging nouns and adjectives in the tweets. Nouns are used by the Feature Finder module to extract features. Adjectives are used by the Polarity Discovery module for opinion mining. It uses the Stanford trained tagger to perform its function.

6) StopWords class

This class contains a list of English stop words. Stop Words are words which do not contain important significance as far as product feature extraction and opinion mining is concerned. These words are filtered out from messages because they return vast amount of unnecessary information. The POSTagger class uses this class to improve its Tagging process. If a tagged noun or adjective is in a list of stop words, it is discarded.

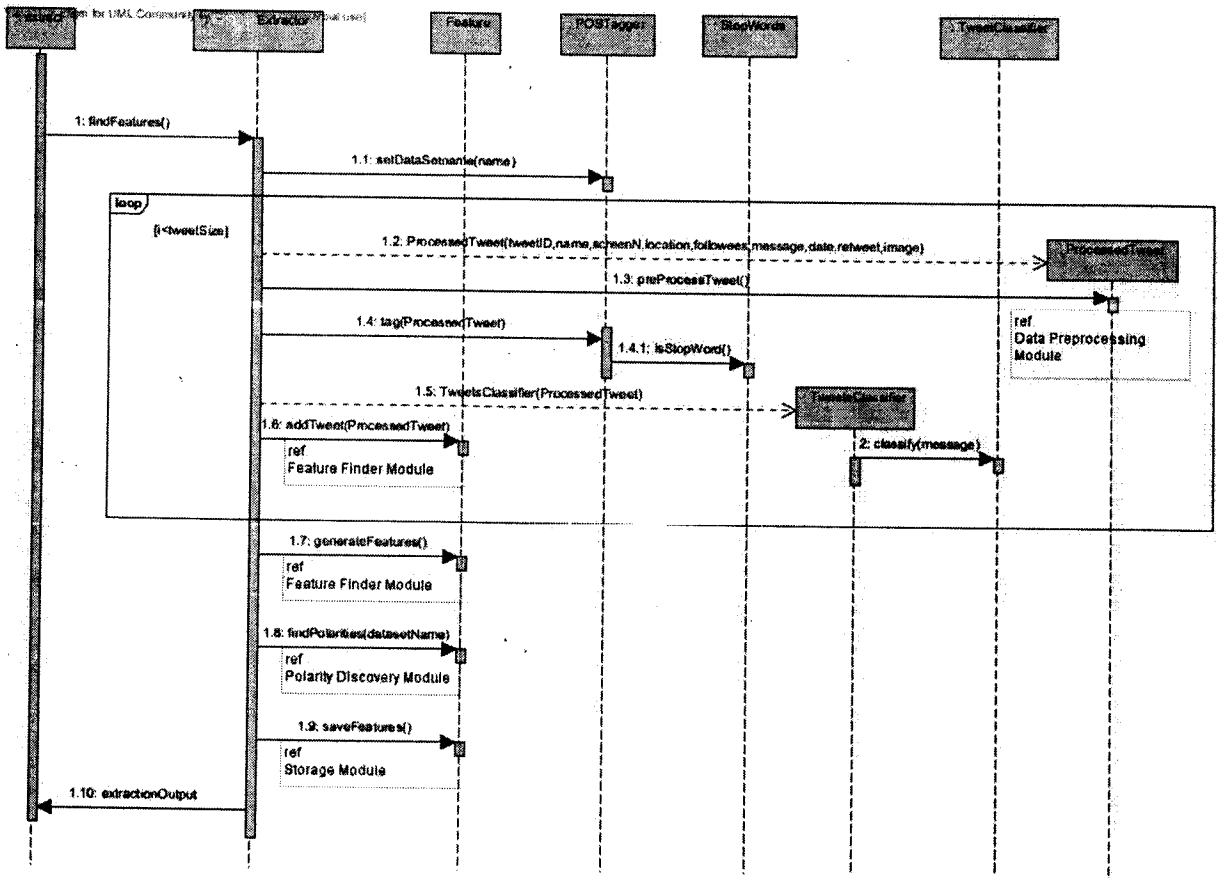

Figure 5.5: Opinion Extractor Sequence Diagram

Figure 5.5 above shows the sequence diagram of the opinion extractor module. The Presentation layer calls the Extractor class to start the feature extraction and opinion mining processes. The Extractor creates the ProcessedTweet class and passes it to the Data Preprocessing module for preprocessing. It then further tags nouns and adjectives in the messages and creates the TweetClassifier objects to classify the messages. It then calls the Feature Finder module to extract features. After the features are extracted, it calls the Polarity Discovery module to discover user opinions on the features. It finally saves the output to the database through the Storage module and displays the output to the user through the Presentation module.

5.1.3 Data Preprocessing Module

The following are some of the characteristics of Twitter messages;

I. Grammatical Errors

   a. Colloquial style: tweets also feature a colloquial style which can be described as "I write the way I speak"; this includes all the phrasing, swear words, chopped words, etc which are typical of spoken language. This introduces a lot of grammatical errors.

   b. Lack of punctuation marks: in order to save characters, users write tweets without punctuation marks. This introduces a lot of grammatical errors.

II. Length: The maximum length of a Twitter message is 140 characters.

III. Use of tokens: Tweets include special tokens such as @ for user names, # for trending topics; they also have http links for related content

IV. Repeated letters: Tweets contain very casual language. For example, hungry can be written as huuuuuungry, or hunnnngry.

V. Misspellings: Twitter users post messages from many different media, including their cell phones. The frequency of *misspellings* in tweets is much higher than in other domains.

VI. SMS style: Tweets have also adopted the emoticons, abbreviations, slang, etc, which is typical of SMS. (E.g. smh -> shaking my head, <3 love, etc)

These characteristics make the Twitter messages informal and hard to process. This module is responsible for transforming an informal Twitter message into a formal one. This process is

called data preprocessing in this project. To achieve this function, the module has the following classes as shown in the class diagram of Figure 5.6 below;

1) Preprocessor class

This class removes usernames in messages, removes hash tag (#) symbol in the messages, removes URLs in the messages, remove the Twitter keyword 'RT' from the messages and remove emoticons from the message. An emoticon is a met-communicative pictorial representation of a facial expression which in the absence of the prosody serves to draw a receiver's attention to the tenor or temper of a sender's nominal verbal communication e.g. ☺.

2) RepeatedCharacterReplacer class

It detects repeated characters in words and corrects them to the correct number of characters. It achieves this function by using the English dictionary.

3) EnglisSpellChecker class

This class represents the English dictionary. It corrects spelling errors and checks if a word is an English word or not.

4) SlangSpellChecker class

Expands slang words into their actual meaning. Slang is the use of informal words and expressions that are not considered standard in the speaker's language or dialect but are considered acceptable in certain social settings. Example '2b' means 'to be'. To achieve this, it uses a Slang dictionary which was compiled in this project. The slang dictionary used in this project is found in Appendix 1.

5) GrammarCorrector class

This class Corrects spellings and grammar errors in the messages. It uses the Language Tools library to perform its function.

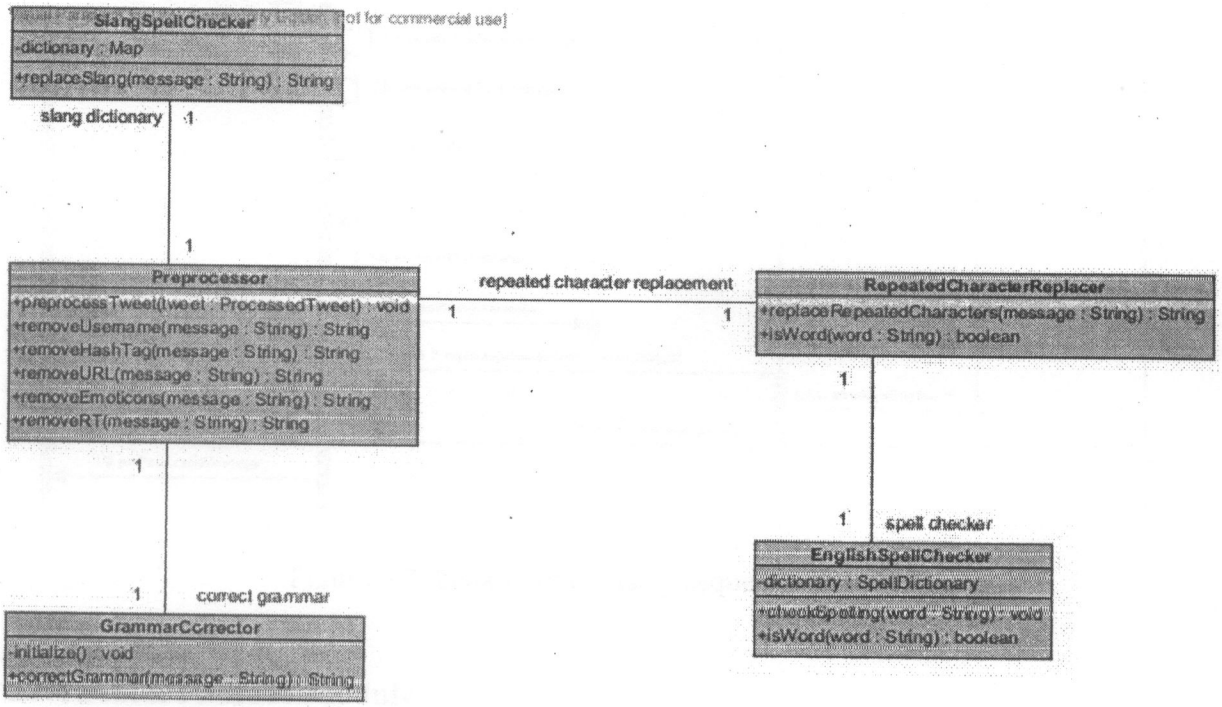

Figure 5.6: Data Preprocessing Class Diagram

Figure 5.5 below shows the sequence diagram of the Data Preprocessing module. The InsightsExtractor class from the Insights Extractor module given in section 5.2 of this chapter calls the Preprocessor class to start the data preprocessing process. The Preprocessor class removes from the messages; user names, hash tag symbol, URLs, emoticons and 'RT' keyword. It then calls the RepeatedCharacterReplacer object to correct repeated characters in words. The Preprocessor object then calls the GrammarCorrector class to correct the grammar in the message. The diagram on the next page shows the details of this process.

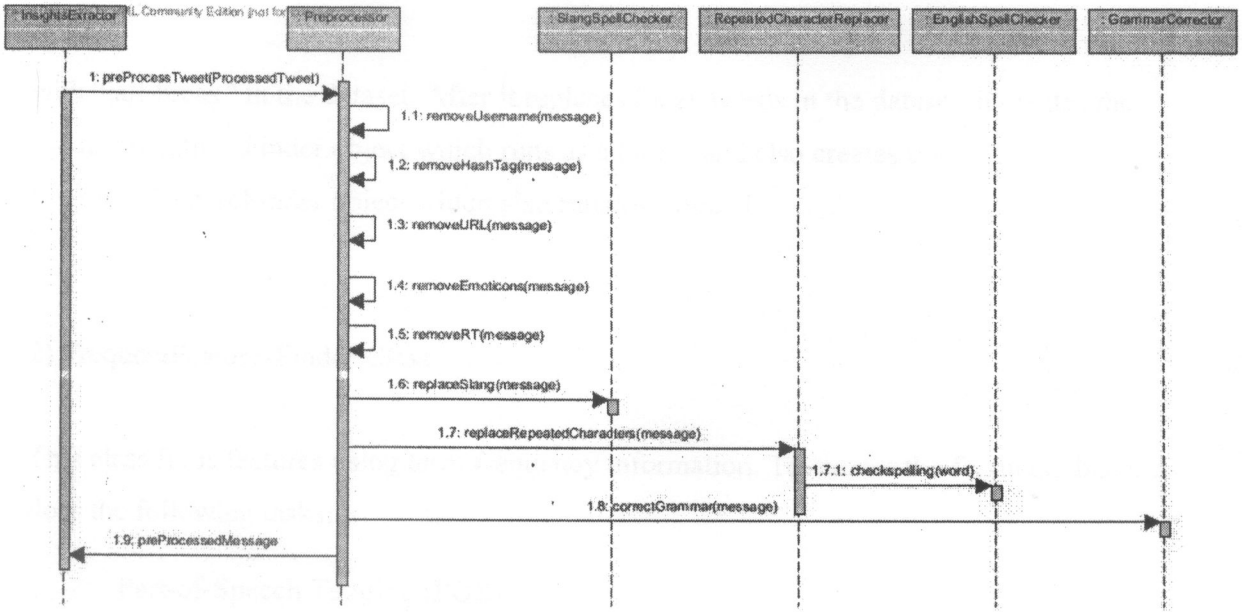

Figure 5.7: Data Preprocessing sequence diagram

5.1.4 Feature Finder Module

Feature finder module as the name suggests, is responsible for extracting product features from tweets. It employs two techniques; term frequency technique and Meronym patterns. The details of each technique are explained later in this section. To perform the task of feature extraction, this module has the following classes shown in the class diagram of Figure 5.8.

1) ProductFeatureGenerator Class

This class is the controller of this sub-system. It coordinates all the classes of the module. Before extraction begins, this class does fuzzy matching on the data. Fuzzy matching is used to deal with word variants or misspellings. For example, "jailbreak" and "jail-break" actually refer to the same feature. The class first detects all the fuzzy words in the dataset and form fuzzy clusters. A fuzzy cluster is a group of all fuzzy words in the dataset. After fuzzy clusters are formed, the class identifies the most popular fuzzy word in each cluster. The third step is to replace all other fuzzy words in a cluster with the most popular fuzzy word. For example if we have ten "jailbreak" words and twenty "jail-break" Word. All the occurrences of "jailbreak" are replaced

with "jail-break" in the dataset. After it replaces fuzzy words in the dataset, it creates the FrequentFeaturesFinder object which runs as a thread and also creates the MeronymFeatureFinder object which also runs as a thread.

2) FrequentFeaturesFinder Class

This class finds features using term frequency information. To extract the features, this class does the following tasks;

- Part-of-Speech Tagging (POS)
    - Each microblog is parsed and noun and noun phrases are extracted.
- Frequent Features Extraction
    - This task is concerned with extracting frequent product features in microblogs. To do this, association rule mining is used to generate frequent phrases in the microblog dataset. In our project, a frequent phrase is defined as a word or group of words that occur together. So the task here is to generate these phrases that occur very frequent in the given microblog dataset.
    - Association rule mining generates all association rules in the dataset that have support and confidence greater than the user specified minimum support and minimum confidence.
    - How do we extract frequent phrases using Association mining? The system uses the Apriori algorithm on the tagged nouns in the dataset to extract frequent nouns. The Apriori run in two phases, the first phase is to find frequents itemsets in the dataset according to the given minimum support and then it generates rules from the frequent itemsets. In this project, we are only interested in the first phase of the algorithm. We run the Apriori Algorithm on the dataset to extract frequent nouns that appears with a minimum support of 0.01 which is 1%. These nouns are taken by the system as candidate product features. The algorithm can produce many nouns with different number of words. The default maximum number of words a feature can contain in our system is 5 but the user can increase and reduce depending on his needs.

- Feature Pruning
    - Some of the frequent features produced by the Apriori algorithm and not true features. For example, people on Twitter like using the word 'ass' a lot. Apriori algorithm can detect such a phrase as a feature because it appears many times in the dataset. Feature pruning is a technique that tries to find such uninteresting features and remove them as product features.

1. Compactness pruning: When the Apriori generates frequent features in the dataset, it can bring phrases that contain nouns that never appear together in the dataset. These kinds of features in our project are called features which are not compact. In our system, a feature is compact if the distance between each word in the feature is 3 or less. This is just the default. The user can reduce or increase depending on his needs. These pruning is applicable to product features that has two or more nouns or words. If a feature has distance between its words 3 or less in at least three tweets, then it is considered as a compact feature otherwise it is removed from the feature list.
2. Redundancy pruning: Redundant product features are features that appear as subsets of other features. For example, 'phone' can be seen as a subset feature of 'phone charger' feature. The Apriori algorithm produces some features which are subsets of other features. We want to avoid including features which are subsets of other features in our final extracted feature list. How do we know if a feature is a subset of another feature or not? In our system, a feature is not a redundant feature if it appears at least in three tweets alone without any super class feature. What I mean here is; for example if we have a feature 'cable' and in every tweet it appears, a super class feature like 'cable charger' appears also, then 'cable' is a redundant feature and we remove it from the product feature list. But if 'cable' feature appears at least in three tweets alone without any super class, then 'cable' is considered as a product feature. In our system, all features are tested for redundancy except features with the maximum number of words as set by the user.
3. Pruning frequent features using meronym features: features produces by meronym patterns represents Has- A relationship between the product and its features. Hence, the system assumes that every feature produces by the Apriori Algorithm must also

appear in Meronym pattern features. All the features produced by the Apriori Algorithm that do not appear in Meronym pattern features are discarded.

To perform all this tasks, this class uses FrequentFeaturePruner and AprioriAlgorithm helper classes.

3) MeronymFeatureFinder Class

A feature F is a meronym of a product P "if native speakers of English accept sentences constructed from such as P has F or F is a part of P". An example is a phone; it has a battery, case, software and so on. Sentences like a phone has the battery must be accepted by native speakers of English for battery to be a meronym of phone. The system uses this technique to extract meronyms of products in the dataset. This meronyms are taken as candidate features. The patterns used in this system are given below. "P" token matches the product name and "F" token matches a possible feature. The wildcards star (*) token can be any 1 or 2 words. Each message is matched with these meronym patterns; if any of these patterns are matched then F is extracted as a feature. F is only extracted when it is a noun in the sentence.

- P has (*) F
- P with (*) F
- P come(s) with F
- P equipped with F
- P contains(s)(ing) (*) F
- F on (*) P and F in (*) P
- F for (*) P
- P's (*) F
- F of (*) P

To improve the quality of the features produced by meronym patterns, the system employs the following techniques;

1. Using conjunction in meronym patterns: To increase the number of features extracted by the meronym pattern, the class uses conjunction words. Every time the "F" is extracted, this class checks if it is followed or preceded by 'and' or 'or'. If that is the case, the next word, if it is a noun, the class gets it as a feature also.
2. Meronym feature pruning: features generated by meronym patterns are pruned for redundancy as explained above when describing the FrequentFeaturesFinder Class. Also the system gets meronym features that occurs a minimum threshold. The threshold used in this system is 10. Every meronym feature that appears less than ten times in tweets is discarded.

To perform all this tasks, this class uses MeronymPatternMatcherand MeronymFeaturePruner helper classes. Both of the feature extraction methods described above uses nouns in the microblog messages as features. But in some microblogs, the same features are used as verbs and pronouns. To overcome this problem, the system uses the following technique;

Product feature expansion: Using the identified features, the system goes in those microblogs without features and stem words using the Stemmer class. Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form—generally a written word form. Every stemmed word is compared with the generated features. If a word is similar to any feature, it is taken as a feature it is similar to.

The InsightsExtractor class from the Insights Extractor module calls the ProductFeatureGenerator class to start the feature extraction. The ProductFeatureGenerator class performs this function by calling FrequentFeaturesFinder class and the MeronymFeatureFinder class. The FrequentFeaturesFinder class calls the AprioriAlgorithm class to generate the frequent itemsets. The MeronymFeatureFinder class calls MeronyPatternMatcher class to match meronym patterns in microblog messages. MeronymFeaturePruner class is called to prune meronym features.

After the frequent features are produced, ProductFeatureGenerator class calls the FrequentFeaturesPruner to prune the features.

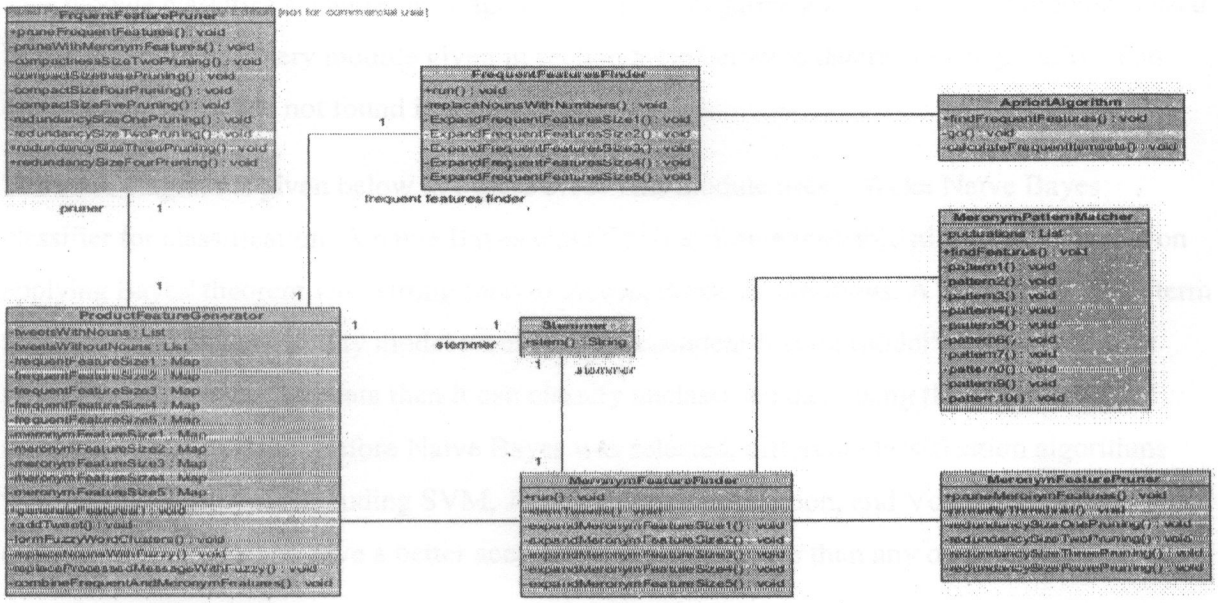

Figure 5.8: Feature Finder class diagram

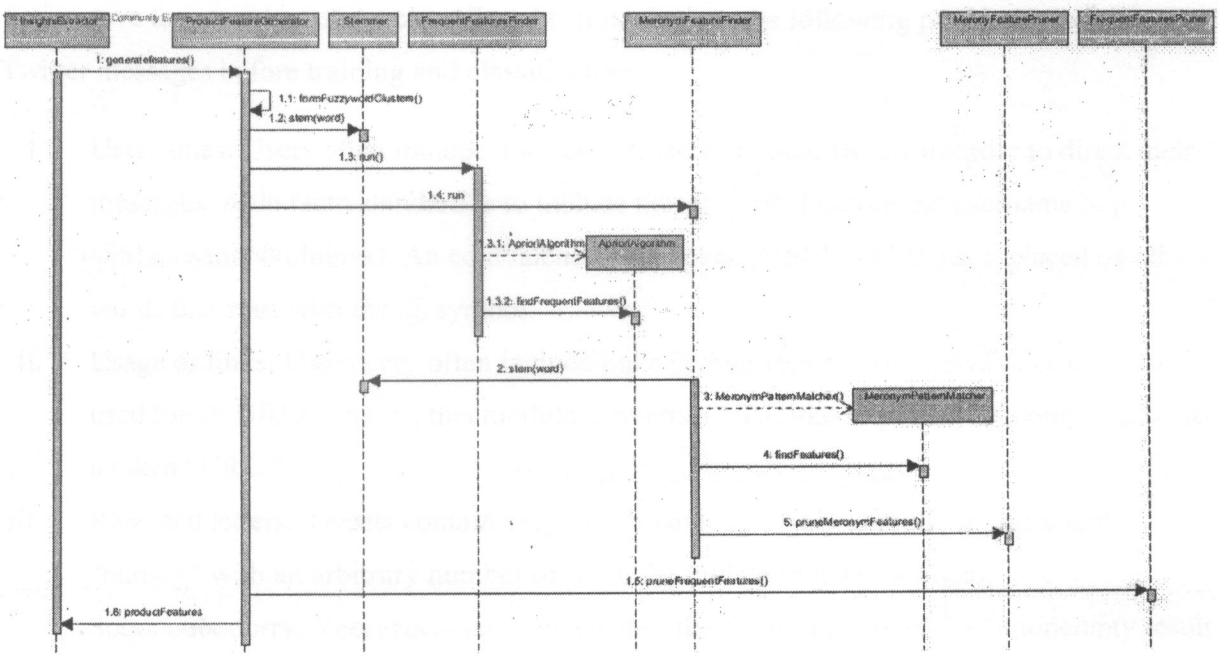

Figure 5.9: Feature Finder sequence diagram

5.1.5 Tweet Sentiment Classifier Module

This module classifies a Twitter message into positive, negative and neutral. This module is used by the Polarity Discovery module given in section 5.1.6 below to determine the polarity of an opinion word which is not found in the Lexicon.

The class diagram is given below in Figure 5.10. This module uses a Weka Naïve Bayes classifier for classification. A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model". The classifier is trained on some classified data then it can classify unclassified data using the knowledge acquired during training. Before Naïve Bayes was selected, different classification algorithms were tested, algorithms including SVM, J48, Multilayer Perception, and Voted Perception algorithms. Naïve bayes gave a better accuracy and performance than any other Algorithm.

To classify a Twitter message, the saved trained classifier is retrieved from the disk and loaded into memory. For each message, it is called to classify it and returns a class label (positive or negative).

To improve the accuracy of the classifier, this module does the following processing on the Twitter messages before training and classification.

I. 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. @MapusanoNkululusa). An equivalence class token (USERNAME) is replaced on all words that start with the @ symbol.

II. Usage of links: Users very often include links in their tweets. An equivalence class is used for all URLs. That is, this module converts a URL like http://tinyurl.com/cvvg9a" to a token "URL."

III. Repeated letters: Tweets contain very casual language. For example, if you search "hungry" with an arbitrary number of u's in the middle (e.g. Goooooooal, Soooooooooorry, Yeeeeeeeeessss) on Twitter, there will most likely be a nonempty result set. Preprocessing is so that any letter occurring more than two times in a row is replaced

with two occurrences. In the samples above, these words would be converted into the token *goal, sorry* and *yes* respectively.

IV. Twitter users like including emoticons in their messages to express emotions. For positive emoticons, an equivalence class token (SMILEHAPPY) replaces all positive emoticons. For negative emoticons, an equivalence class token (SMILESAD) replaces all negative emoticons.

V. Twitter special word RT (which means Retweet) is stripped off from the messages.

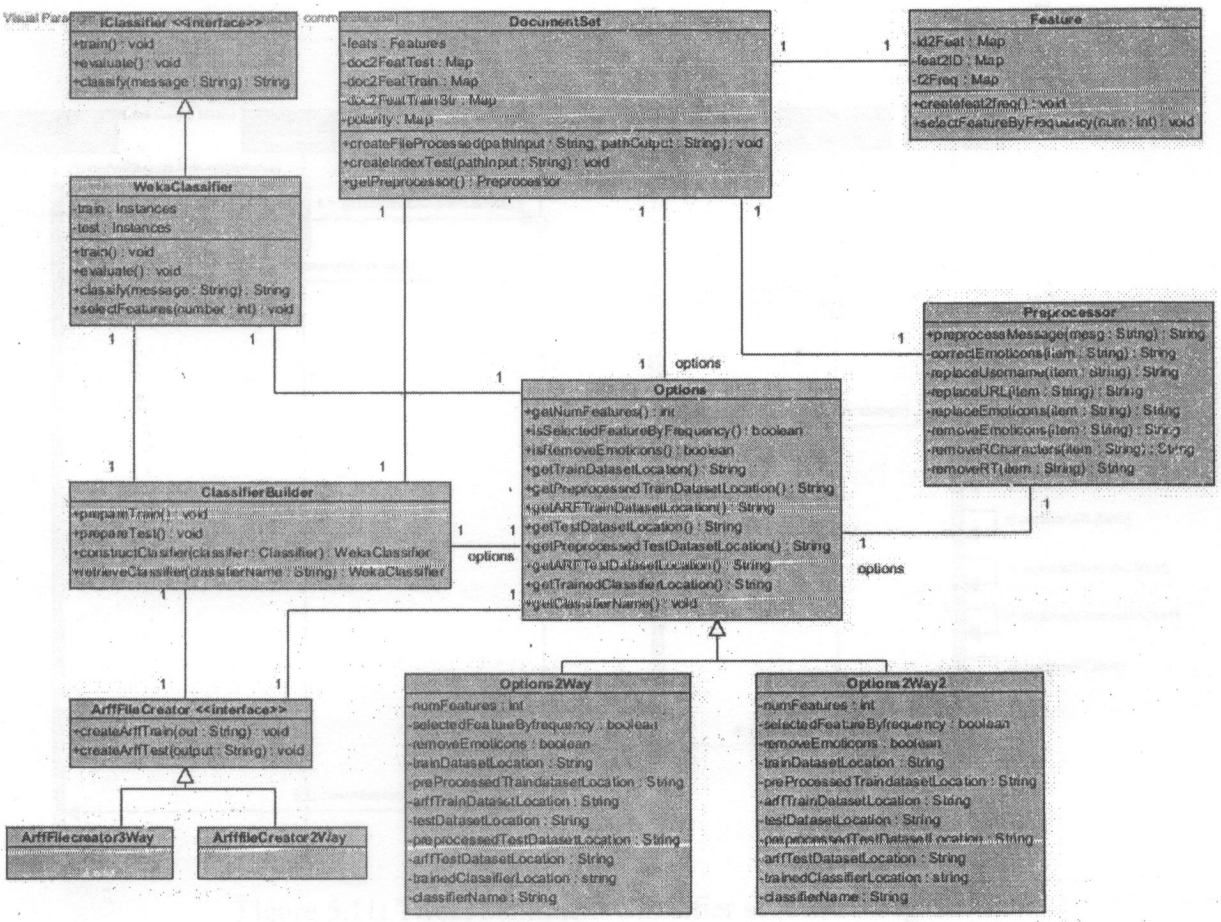

Figure 5.10: Tweet Sentiment Classifier class diagram

Figure 5-11 below shows the sequence diagram of the Tweet Sentiment Classifier module. The sequence diagram only shows the procedure for classification. It does not show the training procedure. To classify a tweet, the ClassifierBuilder class retrieves the saved trained classifier. It then calls the WekaClassifier class to classify a message. The WekaClassifier class calls the Preprocessor class to process a tweet before classification.

This module is used by the Polarity Discovery module. When the Polarity Discovery Module finds an opinion word that is not in the Lexicon, it calls this module to classify the entire Twitter

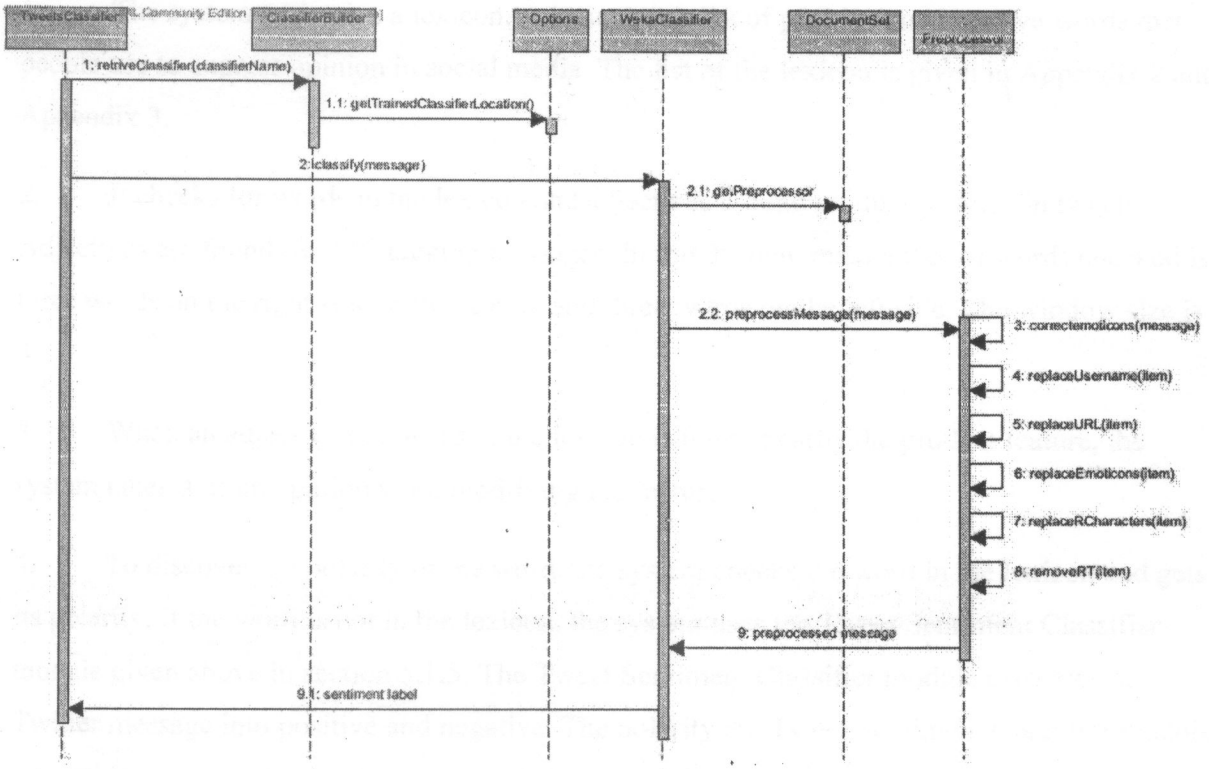

Figure 5.11: Tweet Sentiment Classifier sequence diagram

message. The classification of a Twitter message by this module is taken to be the polarity of the opinion word.

This module is used by the Polarity Discovery module. When the Polarity Discovery Module finds an opinion word that is not in the Lexicon, it calls this module to classify the entire Twitter message. The classification of a Twitter message by this module is taken to be the polarity of the opinion word.

5.1.6 Polarity Discovery Module

This module is responsible for opinion mining. Opinion mining is the process of discovering sentiments users have expressed on product features in the tweets. Figure 5.12 below shows the class diagram of this module. To perform its task, this module employs the following techniques;

1. The system maintains a lexicon. A lexicon is a list of positive and negative words that people use to express opinion in social media. The list of the lexicon is given in Appendix 2 and Appendix 3.

2. It checks for words in the lexicon and adjectives around product features in tweets. Adjectives are found by POS tagging messages. In this system, the number of words checked is three words on the right side of the feature and three words on the left side. The window size is 3.

3. When an adjective or a word in the lexicon is found nearby the product feature, the system takes it as an opinion word modifying the feature.

4. To discover the polarity of the word, the system checks the word in the lexicon and gets its polarity. If the word is not in the lexicon, the system uses the Tweet Sentiment Classifier module given above in section 5.1.5. The Tweet Sentiment Classifier module classifiers a Twitter message into positive and negative. The polarity the Tweet Sentiment Classifier module returns, is it assumed to be the polarity of the opinion word

5. If there is a negation word between a feature and the opinion word, the polarity of the opinion is reversed. The negation words used in this system are; not, however, but, despite, though, except, although, oddly, aside, wont, doesn't and cant.

6. The newly discovered adjectives are saved in the lexicon in the process called lexicon expansion. This way, a lexicon will be growing as the system runs which consequently improves the performance of the system

7. In tweets where they are no features, the system assumes that the feature is the product name itself. Hence opinion mining is done around a product name.

For each product feature, the module counts how many Twitter messages are positive, negative and neutral. An opinion word is judged positive and negative using the Lexicon dictionary. If there is no adjective word near a feature in a message then the system takes neutral as the opinion expressed. This technique works fine for most of the cases but in some Twitter messages, this technique fails to work fine. The Opinion mining algorithm given below takes any adjective word or word found in the Lexicon near a feature as an opinion word. This technique has a weakness because some opinion words near features do not modify the features.

The fully dressed opinion mining algorithm is given below. In the pseudo code below, a statement ends with a semicolon. A comment starts with a double slash. Control statements life "if" have an "end" tag.

```
1. Procedure OpinionMining (adjective_list, seed_list, negation_word_list)
2. begin
3.  for each microblog in the dataset;
4.   left_side_opinion_word; // Opinion word on the left
       side of the feature
5.   right_side_opinion_word; // Opinion word on the right
       side of the feature
6.   //Identify the opinion word
7.   if( microblog contains a feature)
8.    indentify a nearby adjectives and words in the                    seed_list
according to the
```

window size specified by the user both on the left side and right side of the feature;

9. record the nearest adjective or word in the seed_list on the right side in right_side_opinion_word;

10. record the nearest adjective or word in the seed_list on the left side in left_side_opinion_word;

11. end if;

12. //Judge the polarity of the opinion word

13. if(right_side_opinion_word is not empty and left_side_opinion_word is not empty)

14. if((right_side_opinion_word distance from the feature is equal to left_side_opinion_word distance from the feature)

15. left_side_opinion_word is taken as the opinion word for the feature;

16. if(left_side_opinion_word is in seed_list)

17. get the opinion polarity from the Lexicon;

18. else

19. judge its polarity using a Tweet Sentiment Classifier;

20. add the word in the Extension Lexicon as a new opinion word;

21. end else;

22. check if they is a negation word between the feature and the opinion word.

If yes, change the polarity of the opinion word;

23. else if(right_side_opinion_word distance from the feature is greater than
left_side_opinion_word distance from the feature)
24. right_side_opinion_word is taken as the opinion word for the feature;
25. if(right_side_opinion_word is in seed_list)
26. get the opinion polarity from the Lexicon;
27. else
28. judge its polarity using a Tweet Sentiment Classifier;
29. add the word in the Extension Lexicon as a new opinion word;
30. end else;
31. check if they is a negation word between the feature and the opinion
word. If, change the polarity of the opinion word;
32. else if(right_side_opinion_word distance from the feature is less than
left_side_opinion_word distance from the feature)
33. left_side_opinion_word is taken as the opinion word for the feature;
34. if(left_side_opinion_word is in seed_list)
35. get the opinion polarity from the Lexicon;
36. else
37. judge its polarity using a Tweet Sentiment Classifier;

38. add the word in the Extension Lexicon as a new opinion word;

39. end else;

40. check if they is a negation word between the feature and the opinion

word. If yes, change the polarity of the opinion word;

41. else If(right_side_opinion_word is not empty)

42. right_side_opinion_word is taken as the opinion word for the feature;

43. if(right_side_opinion_word is in seed_list)

44. get the opinion polarity from the Lexicon;

45. else

46. judge its polarity using a Tweet Sentiment Classifier;

47. add the word in the Extension Lexicon as a new opinion word;

48. end else;

49 check if they is a negation word between the feature and the opinion word.

If yes, change the polarity of the opinion word;

50. else If(left_side_opinion_word is not empty)

51. left_side_opinion_word is taken as the opinion word for the feature;

52. if(left_side_opinion_word is in seed_list)

53. get the opinion polarity from the Lexicon;

54. else

55. judge its polarity using a Tweet Sentiment Classifier;

56. add the word in the Extension Lexicon as a new opinion word;

57. end else;

58. check if they is a negation word between the feature and the opinion word.

If yes, change the polarity of the opinion word;

59. else

60. the opinion about this feature is neutral;

61. end for;

62. end

The pseudo code given above is implemented in; FrequentFeatureSize1PolarityFinder, FrequentFeatureSize2PolarityFinder, FrequentFeatureSize3PolarityFinder, FrequentFeatureSize4PolarityFinder, FrequentFeatureSize5PolarityFinder, MeronymFeatureSize1PolarityFinder, MeronymFeatureSize2PolarityFinder, MeronymFeatureSize3PolarityFinder, MeronymFeatureSize4PolarityFinder and MeronymFeatureSiz

Figure 5.13 below shows the sequence diagram of the Polarity Discovery module. The InsightsExtractor class from the Extractor module given in section 5.1.2 of this chapter calls the PolarityFinder class to start the opinion mining. The PolarityFinder class calls DatasetPolarityFinder, FrequentFeatureSize1PolarityFinder, FrequentFeatureSize2PolarityFinder, FrequentFeatureSize3PolarityFinder, FrequentFeatureSize4PolarityFinder, FrequentFeatureSize5PolarityFinder, MeronymFeatureSize1PolarityFinder, MeronymFeatureSize2PolarityFinder, MeronymFeatureSize3PolarityFinder, MeronymFeatureSize4PolarityFinder and MeronymFeatureSize5PolarityFinder classes to find opinion words in the microblog messages

and judge the polarity. To discover opinion polarity, all these classes uses the Lexicon class. During system start up, the Lexicon class loads the lexicon into memory. All newly identified adjectives are added to the Lexicon. To speed up the Opining mining process,

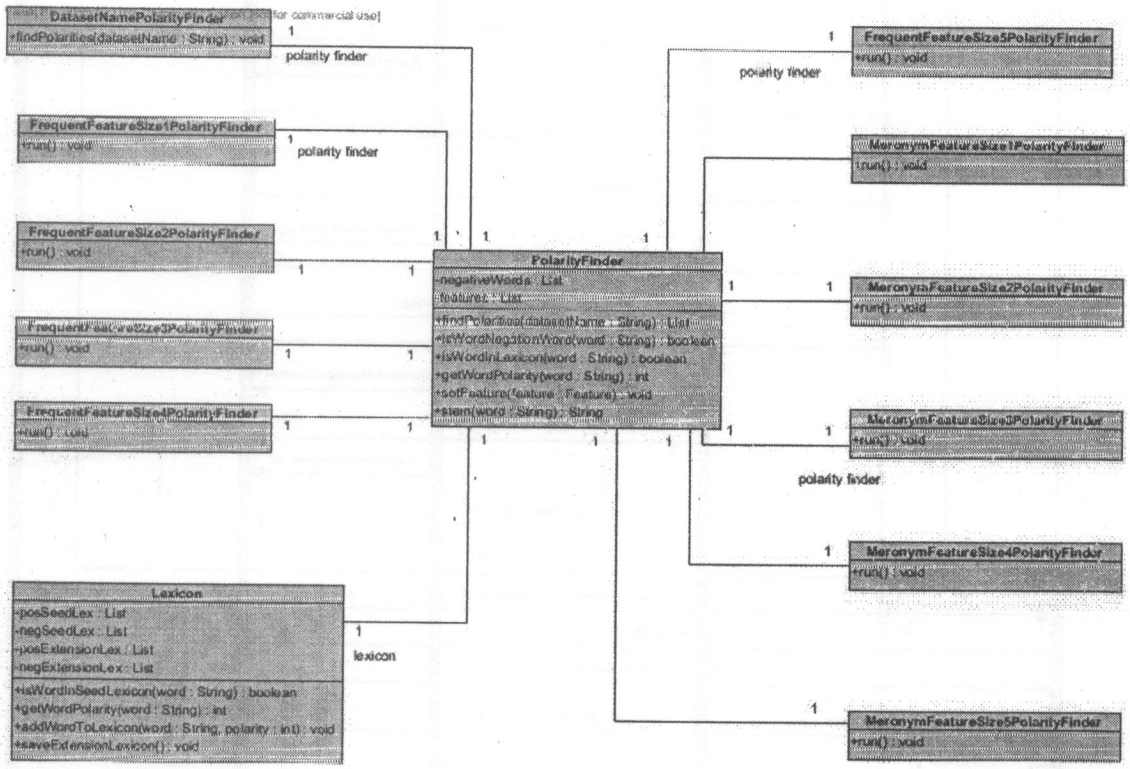

Figure 5.12: Polarity Discovery class diagram

FrequentFeatureSize1PolarityFinder, FrequentFeatureSize2PolarityFinder, FrequentFeatureSize3PolarityFinder, FrequentFeatureSize4PolarityFinder, FrequentFeatureSize5PolarityFinder, MeronymFeatureSize1PolarityFinder, MeronymFeatureSize2PolarityFinder, MeronymFeatureSize3PolarityFinder, MeronymFeatureSize4PolarityFinder and MeronymFeatureSize5PolarityFinder classes are multithreaded. They run in parallel, each class working on a subset of features and a subset of Twitter messages. The PolarityFinder class synchronizes these threads.

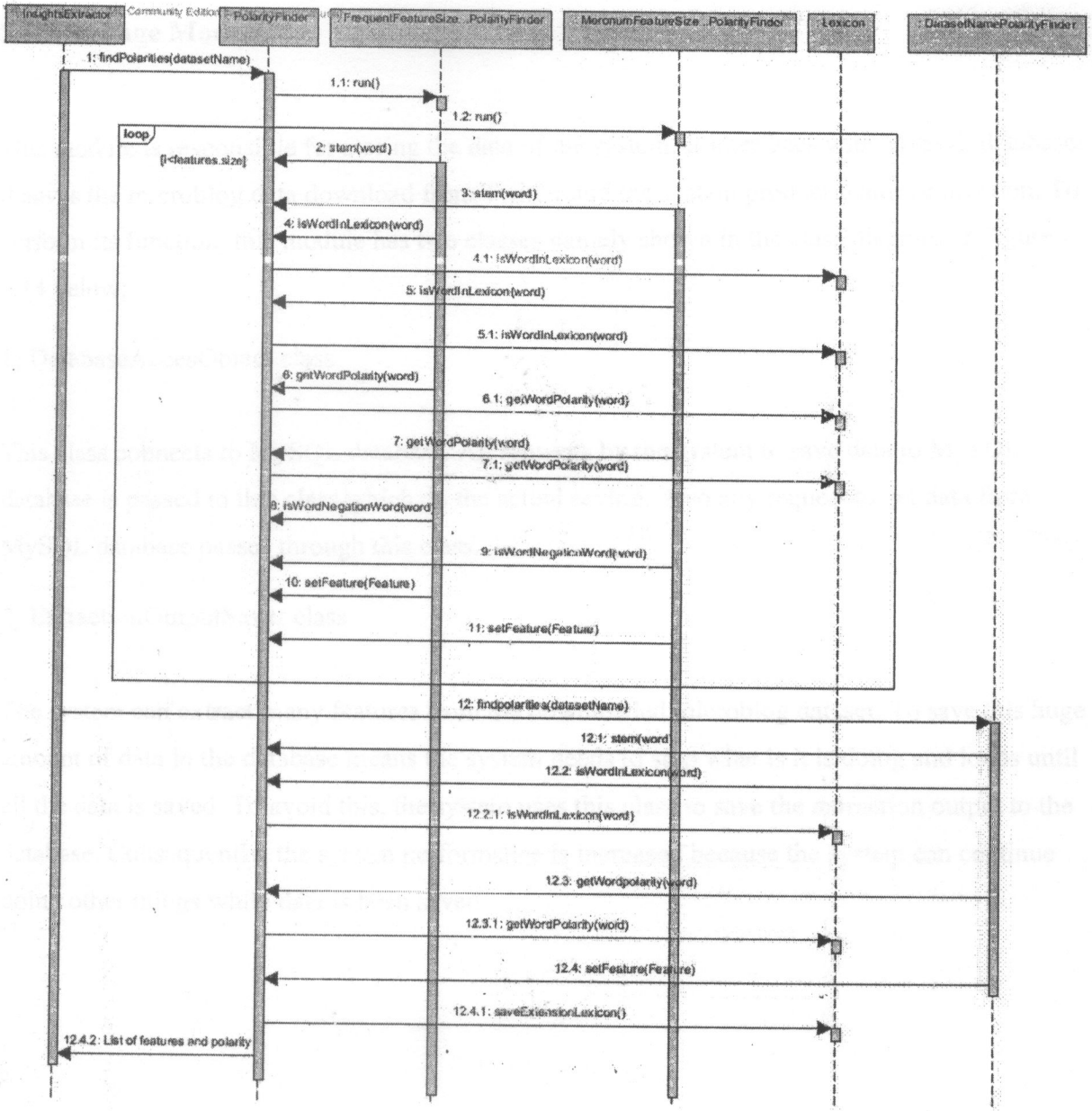

Figure 5.13: Polarity Discovery sequence diagram

5.1.7 Storage Module

This module is responsible for storing the data of the system. It interfaces with MySQL database. It saves the microblog data download from Twitter and the system product feature extraction. To perform its function, this module has two classes namely shown in the class diagram of Figure 5.14 below;

1) DatabaseAccesObject class

This class connects to MySQL database. All requests by the system to save data to MySQL database is passed to this class which do the actual saving. Also any request to get data from MySQL database passes through this class.

2) ExtractionOutputSaver class

The system can extract many features from the downloaded microblog dataset. To save this huge amount of data in the database means the system needs to stop what is it is doing and loops until all the data is saved. To avoid this, the system uses this class to save the extraction output to the database. Consequently, the system performance is increased because the system can continue doing other things while data is been saved.

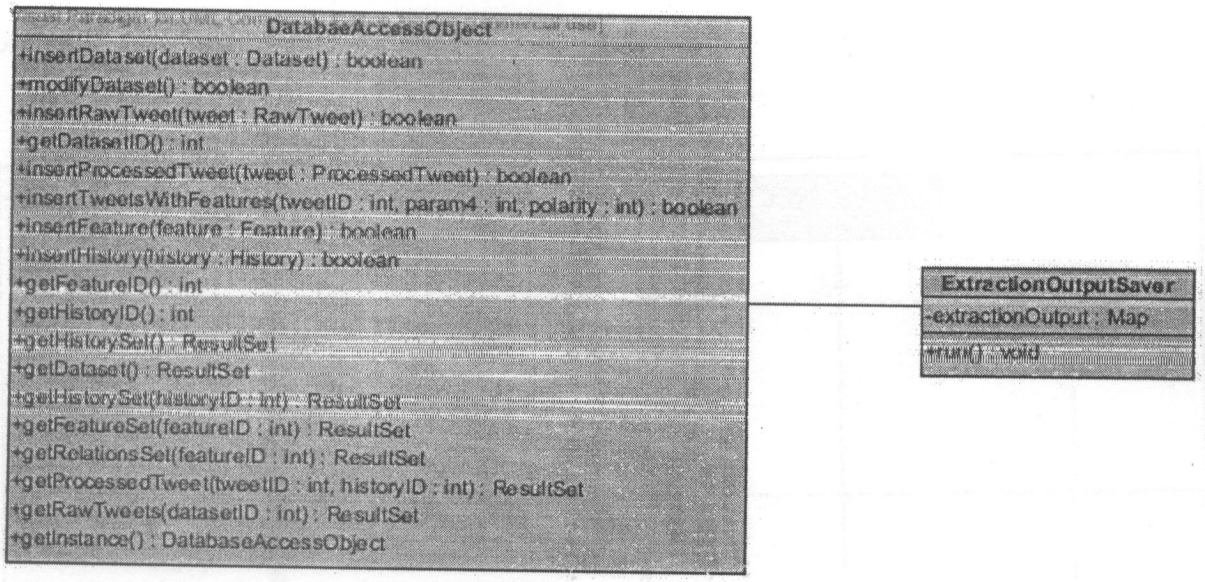

Figure 5.14: Storage class diagram

5.1.8 Presentation Module

The presentation layer is responsible for handling user interactions. It provides a GUI to the user to operate the system. All user interactions are passed to the system logic/model layer. This module is also responsible for ranking both the product features and the tweets before they are displayed to the user. This is the layer which is also responsible for user login in and sessions. The Figure below shows the login sequence diagram.

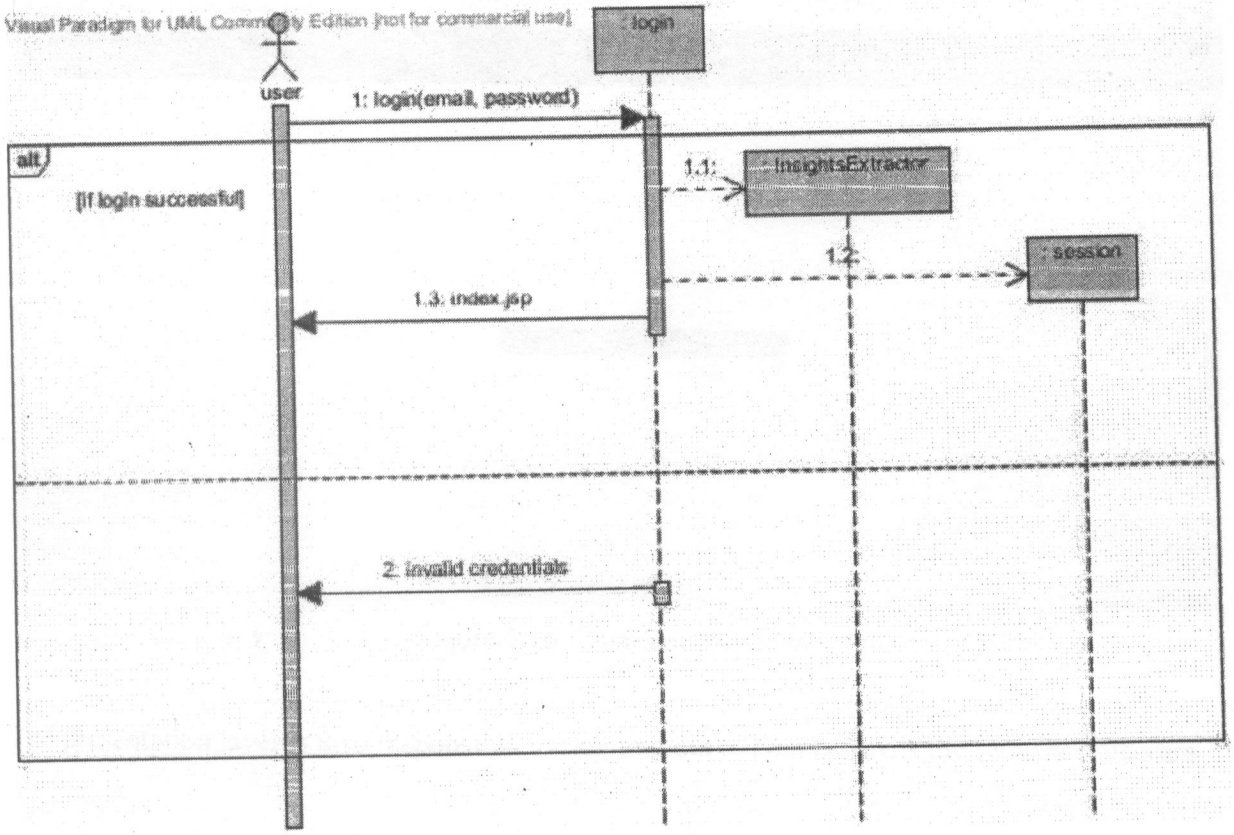

Figure 5.15: Login sequence diagram

User passes in their credentials (email and password) and the login module verifies if there is such a user. if the credentials are valid, the login in module instantiates the insightsExtractor and starts a session for the user. The user is the presented with a home screen as shown in the Figure below. Otherwise, they are presented with an error message telling them that the credentials entered were invalid.

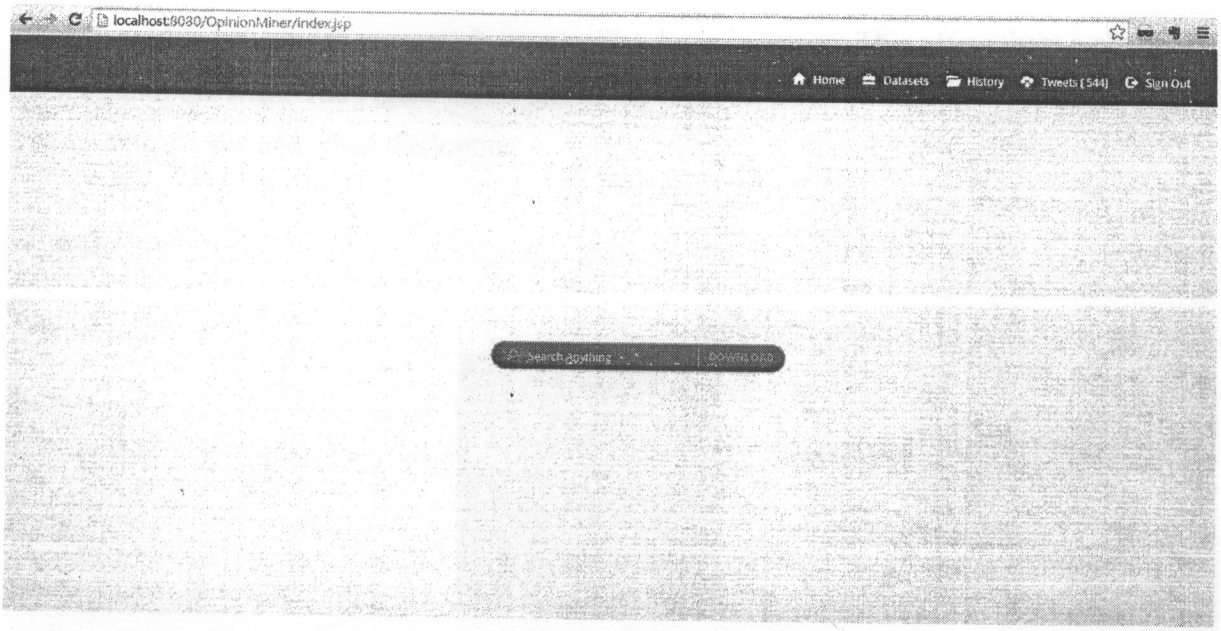

Figure 5.16: User Home Screen

The presentation layer is also responsible for logging out .Given below is the sequence diagram for logout.

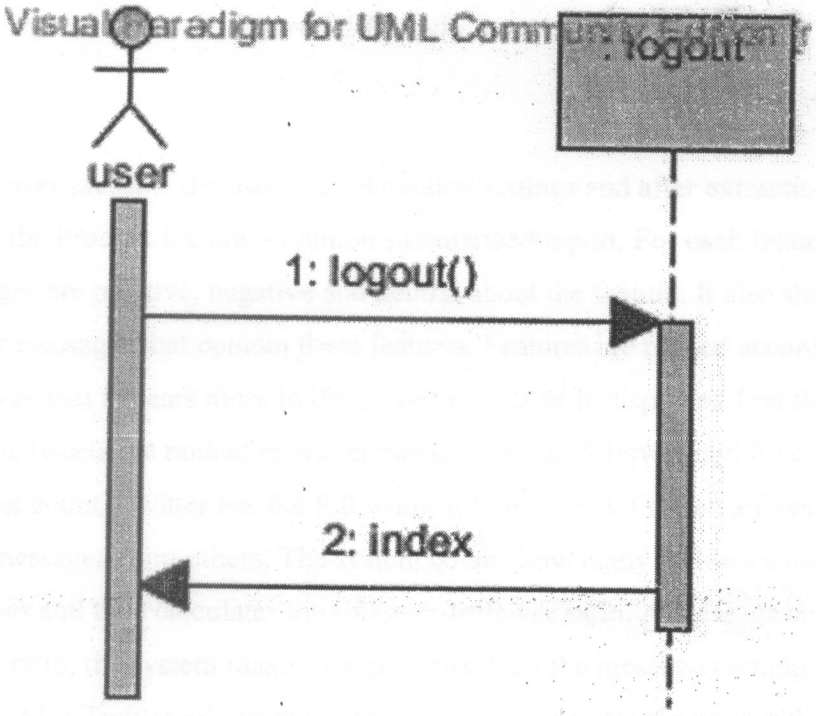

Figure 5.17: Logout Sequence Diagram

When logged out, the index presents the user with a login in screen. This offers them a chance to login again if the wish. The Figure below shows the login screen. This is presented the first time a user visits the site and when the logout.

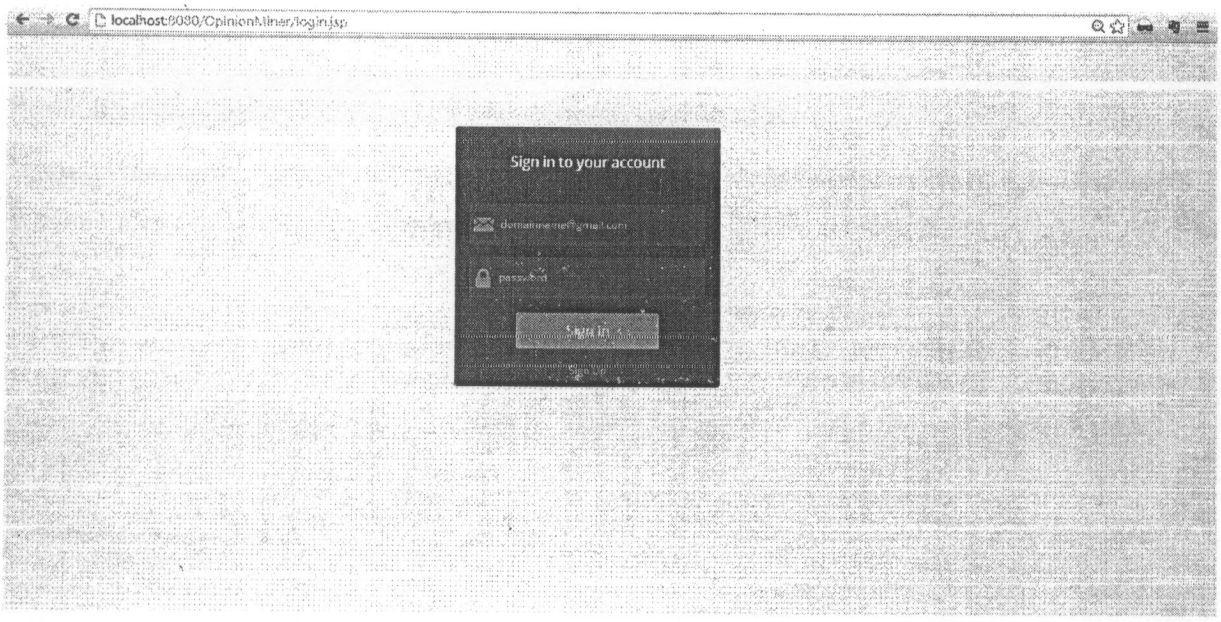

Figure 5.17: Login Screen

The presentation layer presents the user with extraction scttings and after extraction, it presents results displaying the Product feature – opinion summarized report. For each feature, it shows how many messages are positive, negative and neutral about the feature. It also shows the individual Twitter messages that contain these features. Features are ranked according to frequency. A feature that appears more in the messages dataset is displayed first than a feature appearing less. The tweets are ranked based on message sender follower and following counts and message repost count. Twitter has the following-follower model which allows one to follow and receive new messages from others. The system counts how many follows a user and how many a user follows and then calculates the follower-followee ratio. A message coming from a user with a higher ratio, the system makes it appear first than the message coming from a user with a lower ratio. Also Twitter allows message repost which makes message diffuse faster. When a user receives a message from another user, he/she can repost it to his followers. The

system counts how many times a message has been reposted. A message with a higher repost count, the system makes it appear first than a message with a lower count. The detailed ranking algorithm is given in the appendix. The following diagrams are screen shot from the presentation layer showing all the above mentioned details.

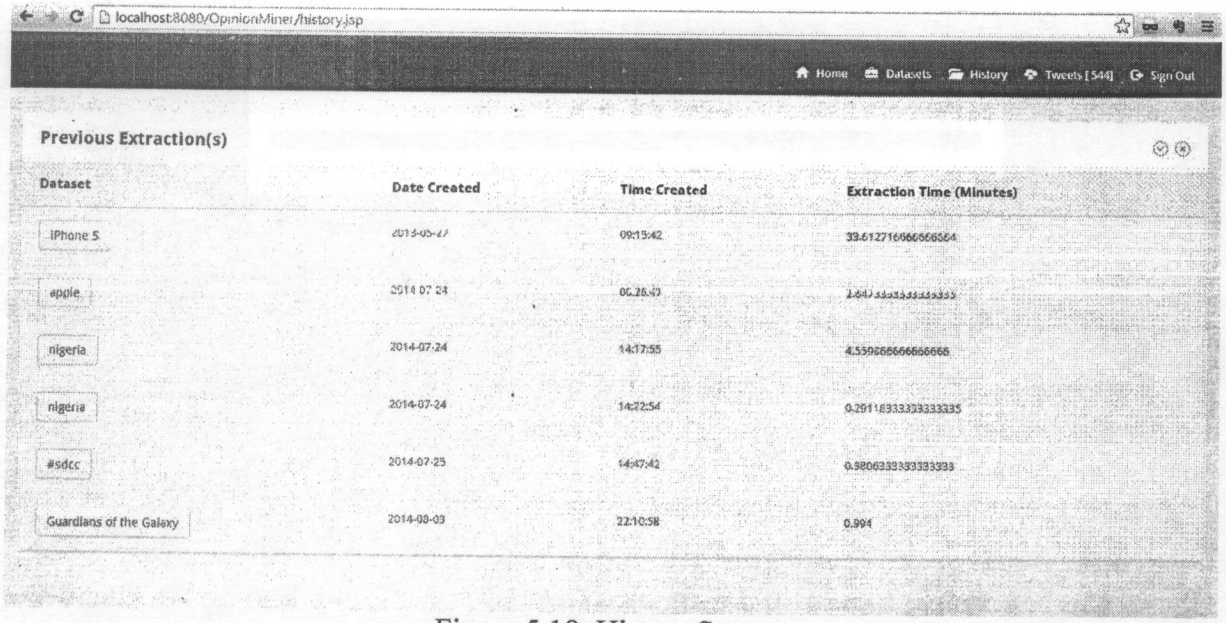

Figure 5.18: History Screen

The history screen shows the dataset that are stored in the system and were extracted previously. The screen has the date and time when a dataset was created and the time it took to extract features and processing opinion mining. A user can then choose to load any of the dataset previously extracted. Figure 5.19 below show how the screen looks like. When loaded, the system displays the results are shown in figure 5.20 below. The feature lie at the bottom of the screen and tweets and opinion mined on these features can be seen by clicking on any feature of interest. The features are extracted according to the preferred settings presented in Figure 5.21 and while the extraction is in process, the user is kept informed on the progress through the extraction progressive bars shown in Figure 5.22. The sub-progress bars are in green represent independent sub process because others run in parallel. The over progress is indicated by the red progress bar at the bottom.

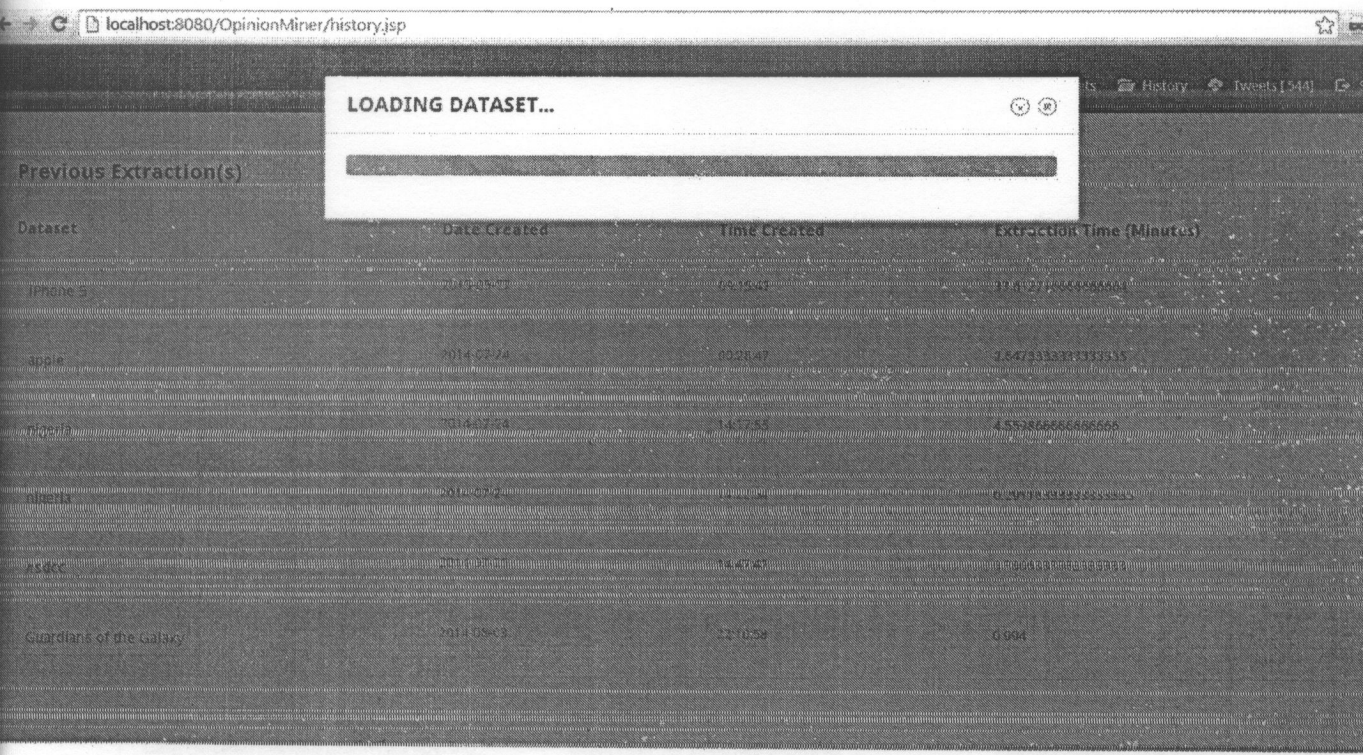

Figure 5.19: Loading Dataset form History Screen

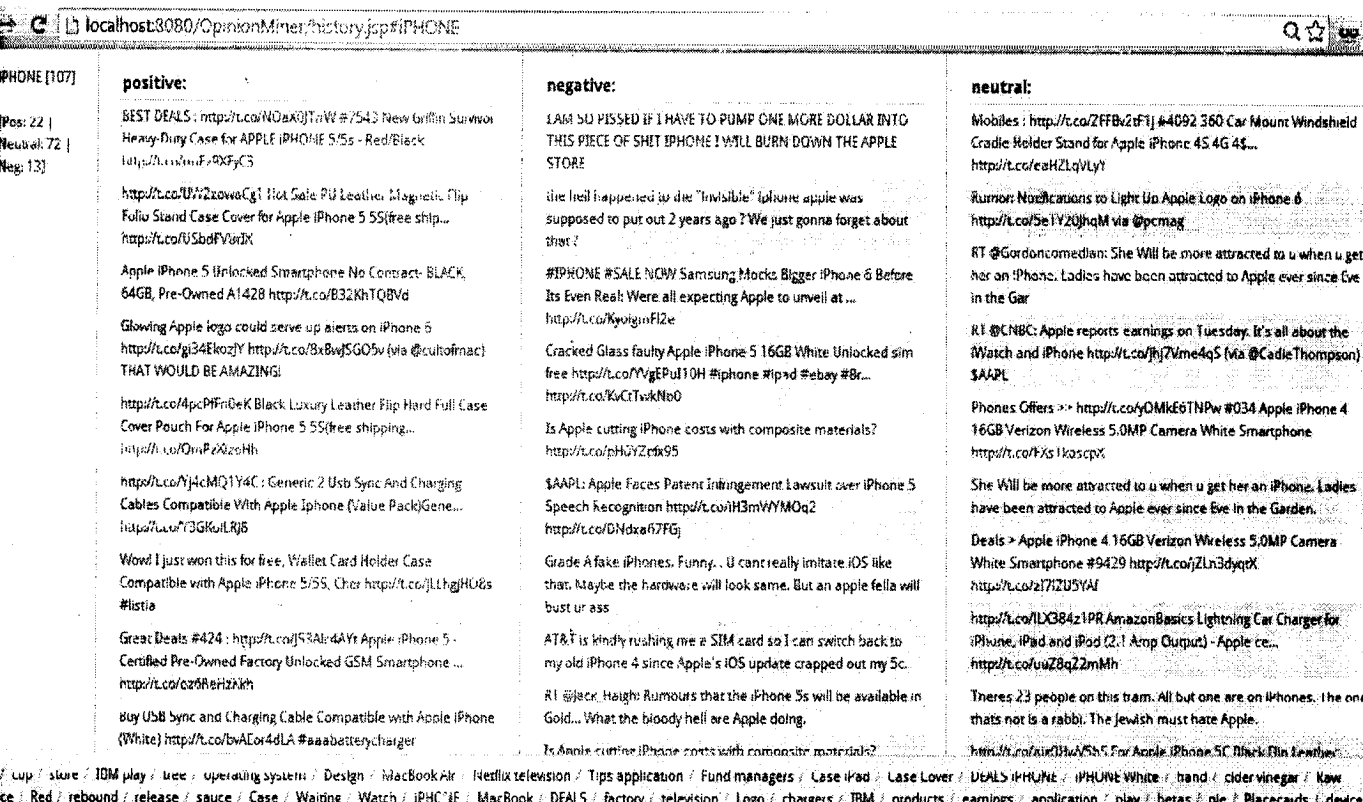

Figure 5.20: Extraction Results View Screen

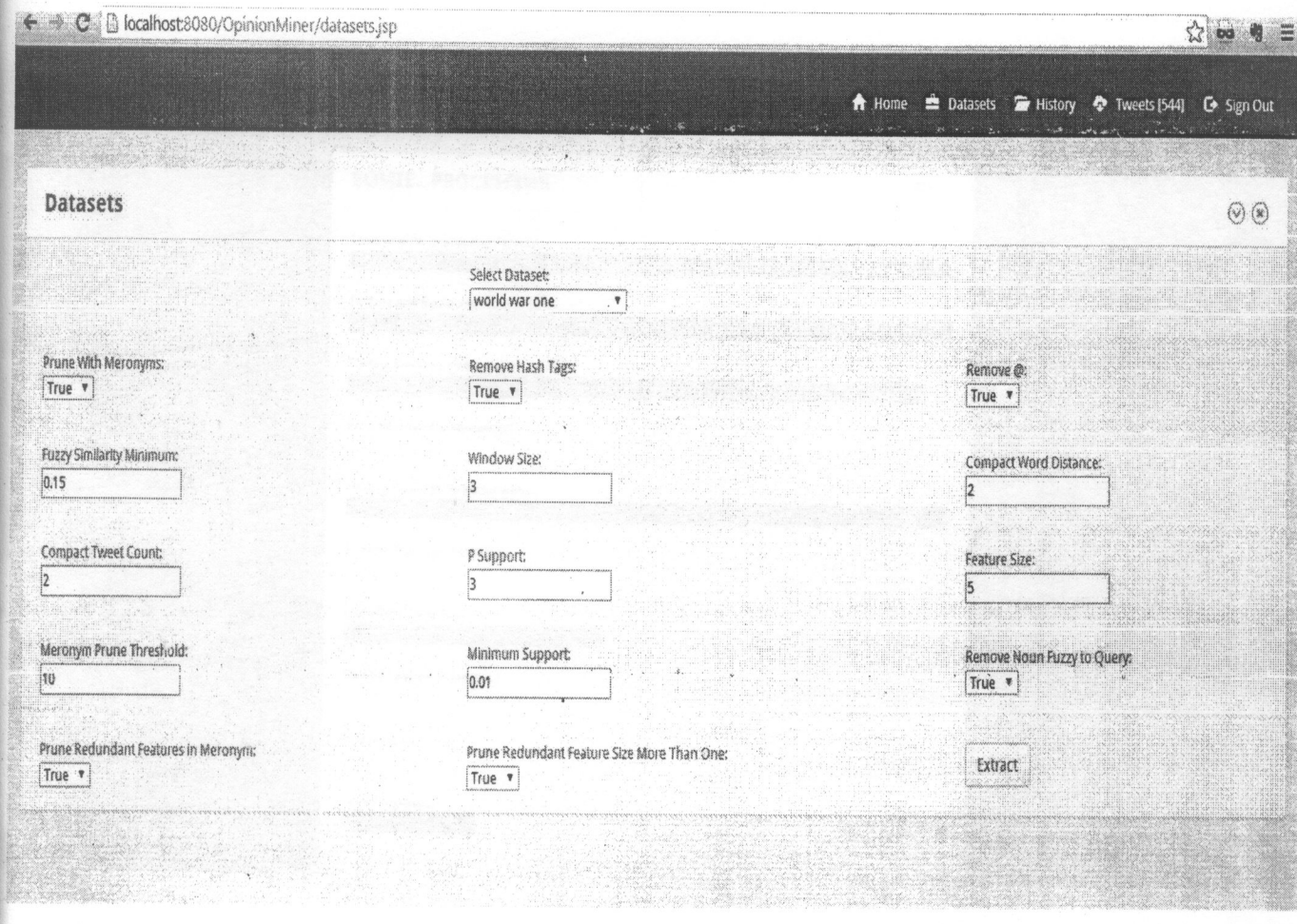

Figure 5.21: Extraction Settings Screen

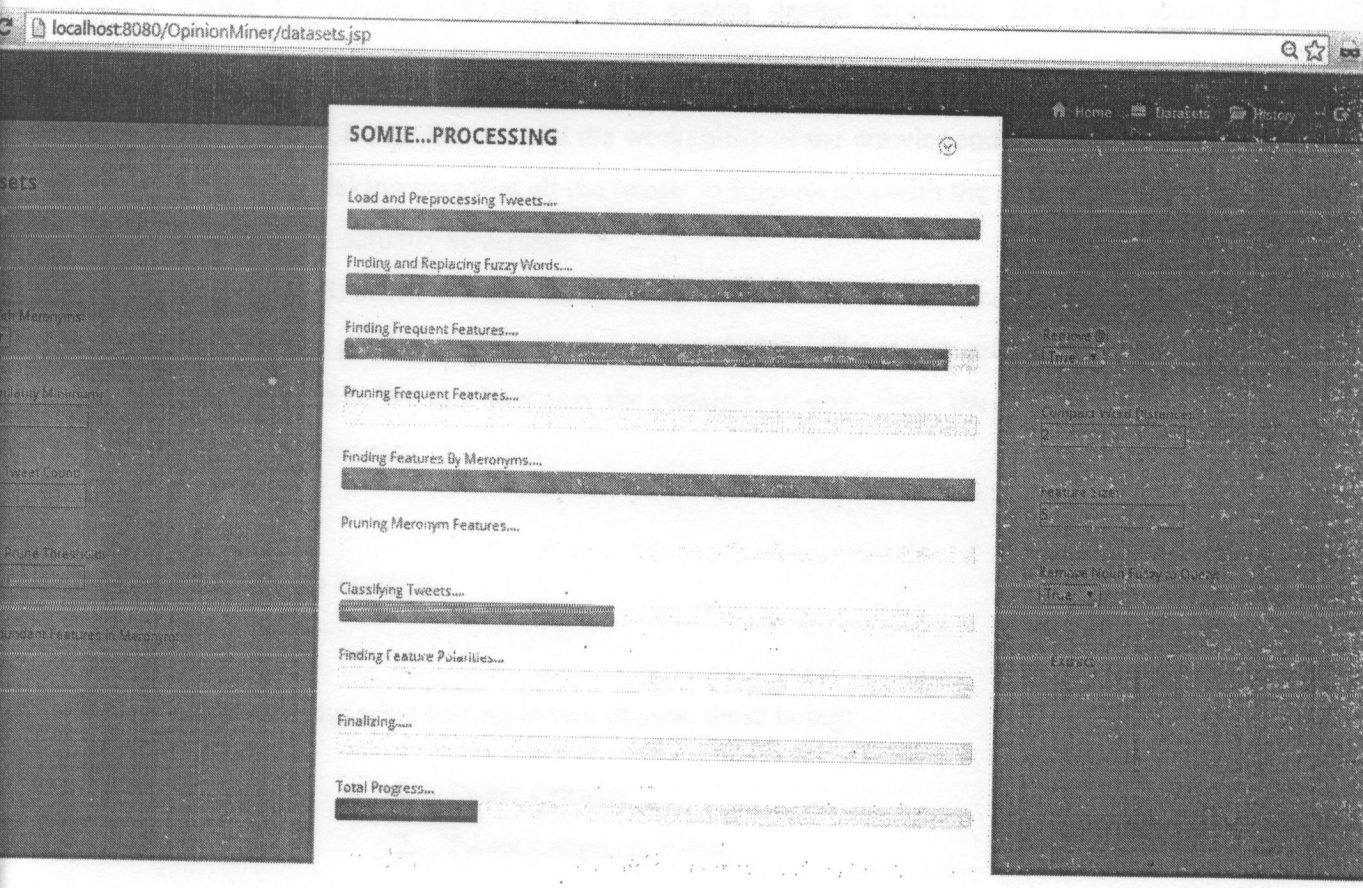

Figure 5.22: Extraction Processing Screen

6.0 TESTING AND VERIFICATION

6.1 System test method

This System had over forty requirements for this project both function and non-functional. The system testing was done according to the system design structure. But only selected 3 components to test these include:-

1. DATA COLLECTION: we test the workability of the crawler and the language filter for the tweets. Making sure that all the tweets in English and with the specified parameter for streaming actually streamed.
2. DATA ANALYSIS: We test the accuracy of the system's discovery of opinion words that modifies product features and opinion word polarity judgment.
3. EASY OF USE: We test how easy the system is to user. From the user interface usage and responsiveness.

6.2 Data collection testing

Data collection testing what testing in two phases, these being:

1. Tweet Crawler
2. Tweet Language Filter

1). Tweet Crawler:

The twitter crawler was developed so as to ensure that tweets are streamed from twitter using the twitter streaming API. The twitter streaming API accepts parameters that enable user to get tweets based on parameters provided by the twitter API.

In order to acquire tweets from twitter, one needs to be familiar with the twitter streaming API and see programming languages that can be used to tap into it. In our case, since we were using java. Twitter provides a library that can be used to tap into and collect tweets has the come in through twitter.

Twitter also needs to authenticate a particular user. A system login credentials need to be obtained from twitter. These include consumer key, secret key, access token and access secret. These four credentials are obtained after creating an online app from the twitter developer website. Table 6.1 below shows crawler testing.

Test case	Scenario	Expected result	Result	Status
tweet stream/downloader	search for topic	streamed tweets	Streamed tweets	Successful

Table 6.1: Crawler Testing

The Figure 6.1 below show a screen shot captured at testing to see if the system is able to downloads tweets in real time. In the figure below, yelp was what was typed in as the search keyword/topic. The tweets appear in a timeline as they are downloaded. To every tweet, the author's image is appended as seen below.

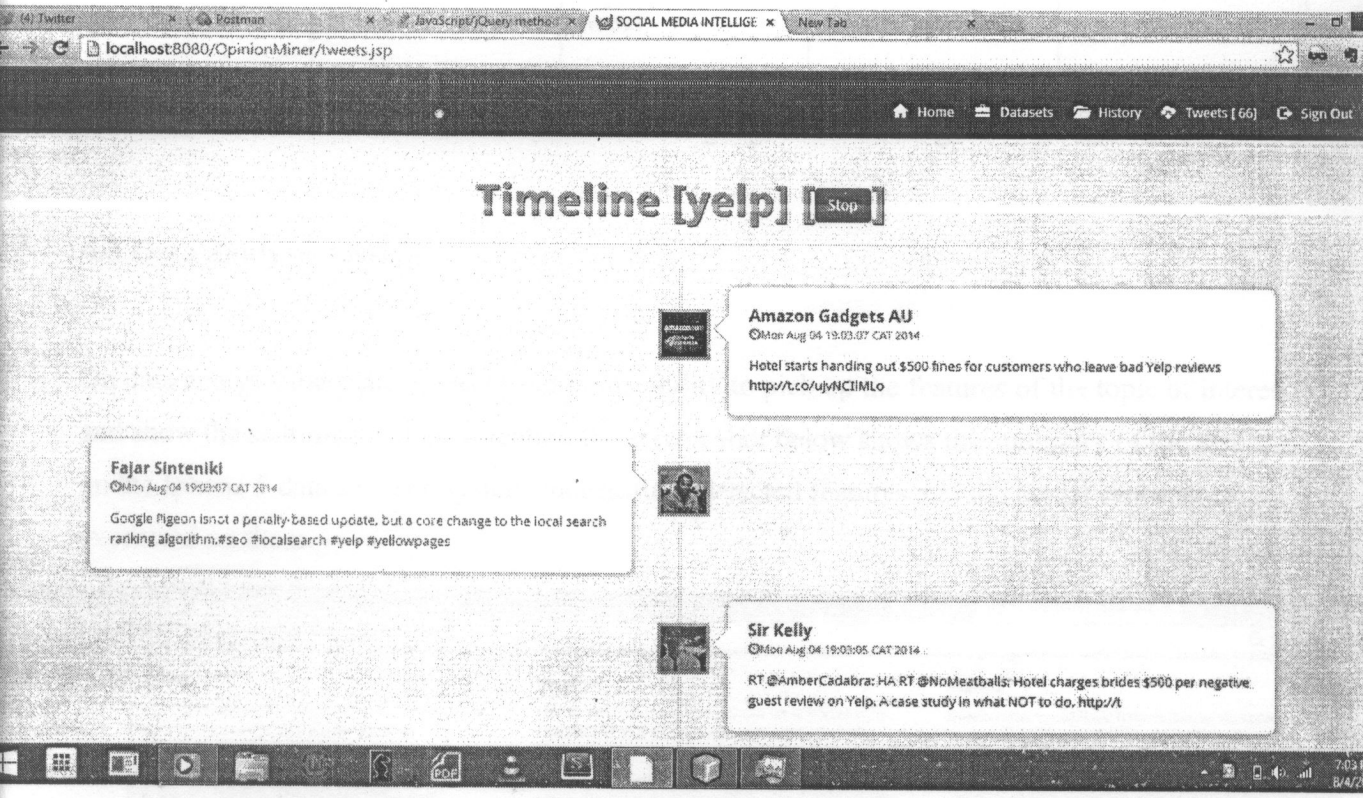

Figure 6.1: Live Streaming of Tweets

2). Language filter:

The language filter used was not the one that of twitter. Actually twitter has a result portion in the JSON result that specifies the actually language which a particular tweet is in. Apparently this was not accurate has users that specify English has their language sometimes can post tweets that may be in another language other than twitter. Therefore we counted the number of English words in tweet. In the number of words in it exceeded a certain percentage the tweet was classified as English.

Test case	Scenario	Expected result	Result	Status
tweet language	English tweet	Accept English tweet	Accepted English tweets	Successful

Figure 1Table 6.2: Language Filter Testing

6.3 Data analysis testing

On data analysis the system was tested on its ability to pick up the features of the topic of interest and show the sentiments of each feature the screen shot below shows the summary of the result on the Iphone 5 data set. The system successfully detected features and the actual sentiments expressed on them.

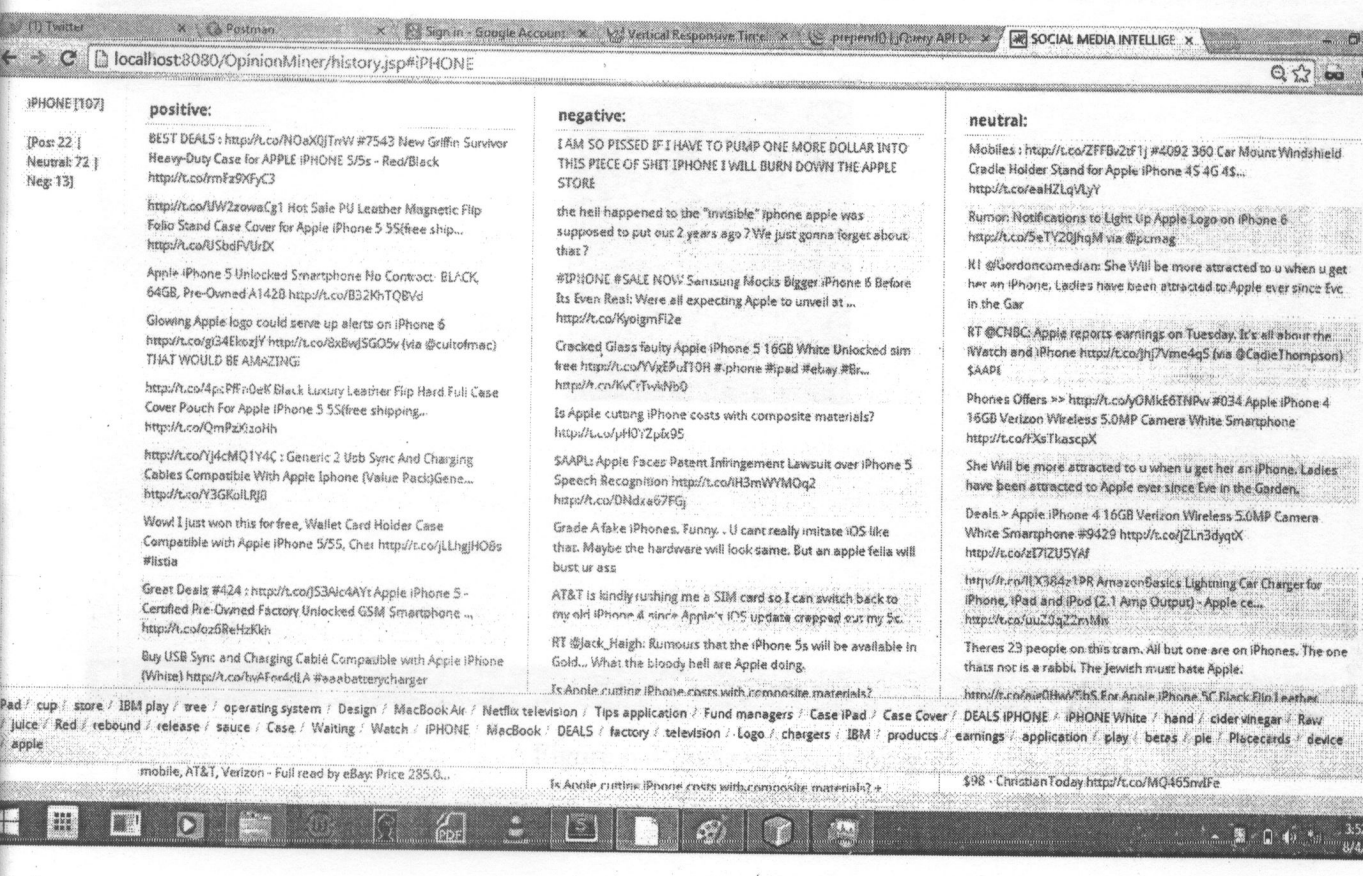

Figure 6.2: Data Analysis results

6.4 Ease of use testing

They system usability was tested from its presentation in terms of graphical user interface interactivity and user aesthetics, table 6.3 tabulates the testing and results.

Test case	Scenario	Expected result	Result	Status
LOGIN	User authentication	Produce home page logged in	Produced home page login in	successful

Table 6.3 Ease of Use Testing

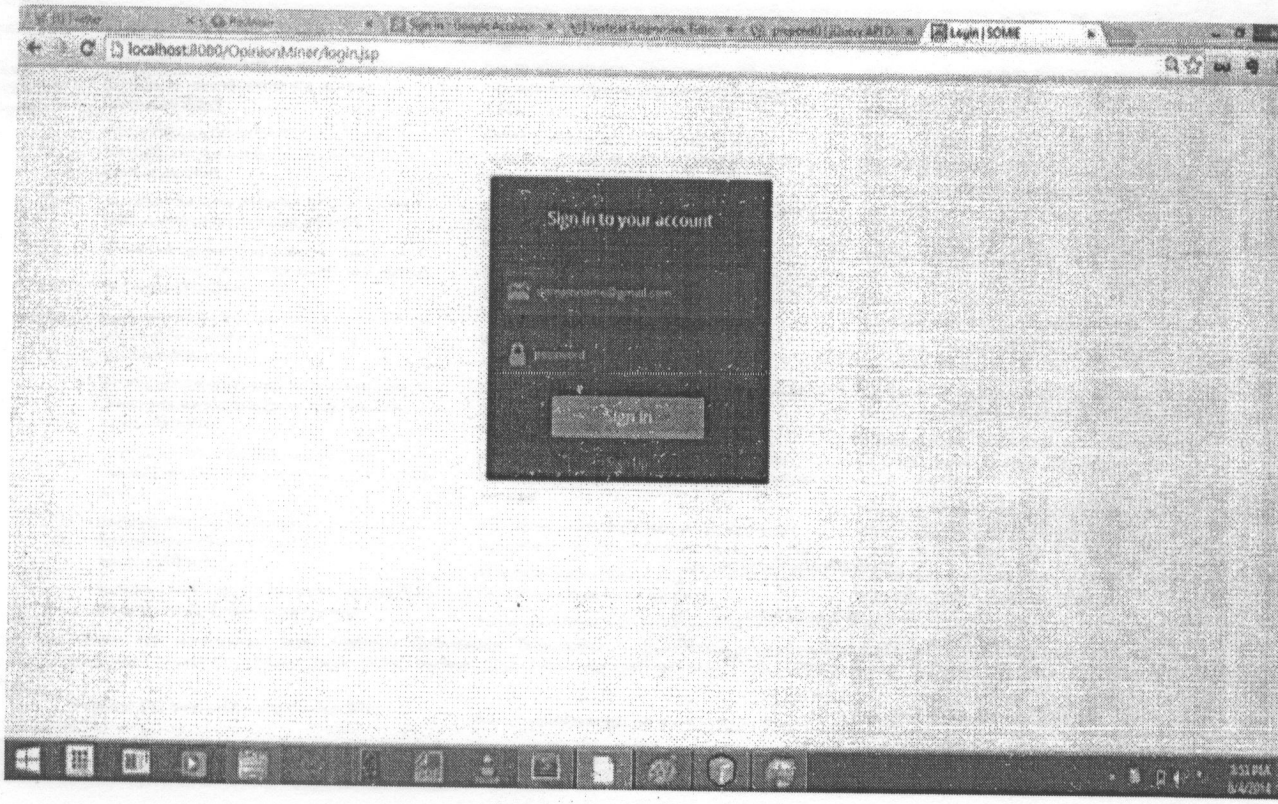

Figure 6.3: Ease of Use testing

6.5 Aesthetics testing

User aesthetics includes presentation beauty and ease of use; this was accomplished using various techniques. Most notably one was in the presentation of the feature extraction report. In the event that the data to display was excessive breadcrumbs where used to enable the user to navigate through a dataset feature sentiments. Screen show below show that this was successful.

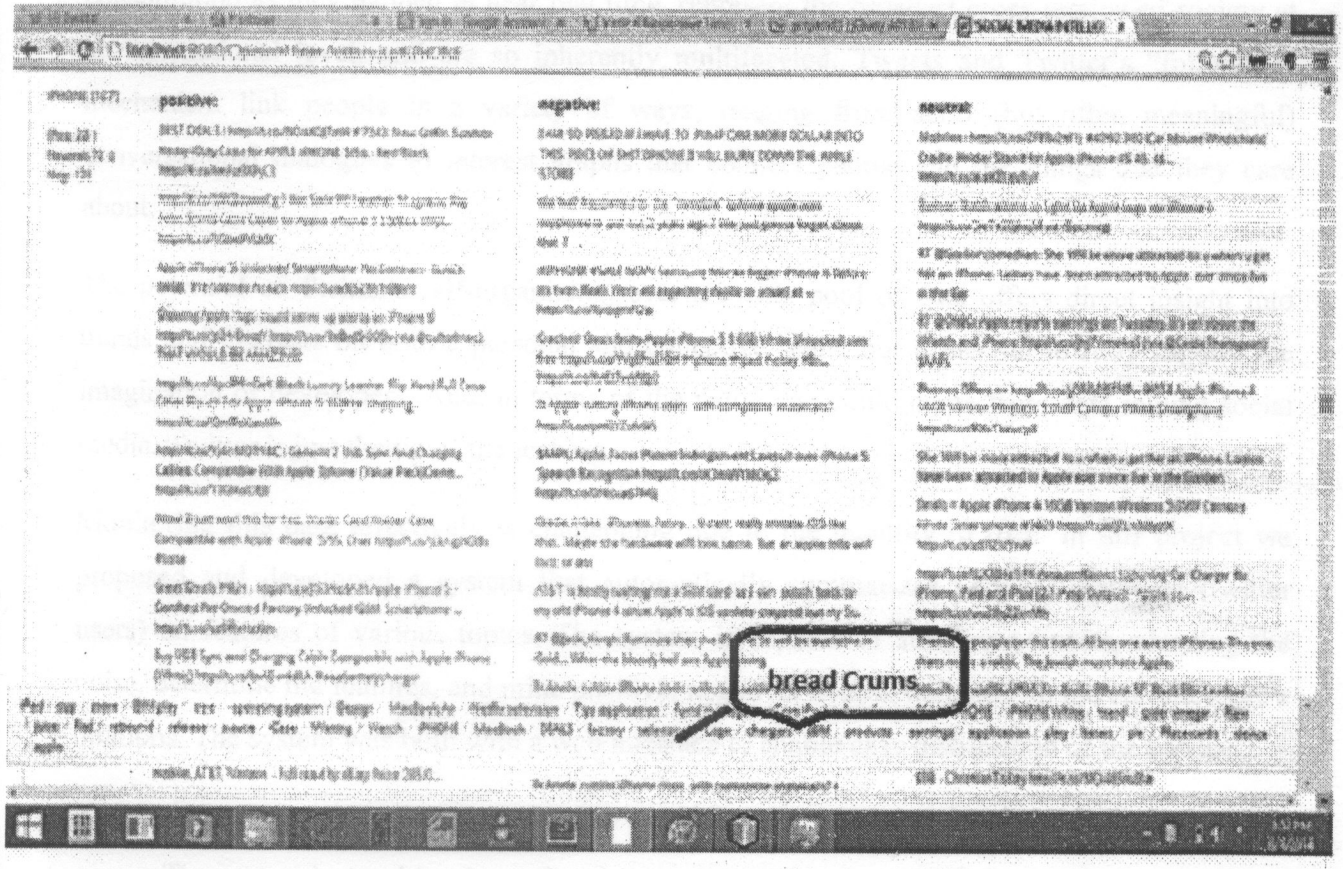

Figure 6.4: Aesthetics testing

7.0 CONCLUSION

With the advent of web 2.0, users have been given the ability to author information and become active participants in the usage of the web. This is truer in social networks than any other kind of media. Social networks produce a large amount of data that could provide insight on various topics of interest. Twitter, a rich source of social data that is a great starting point for social web mining because of its inherent openness for public consumption, clean and well-documented API, rich developer tooling, and broad appeal to users from every walk of life. Twitter data is particularly interesting because tweets happen at the "speed of thought" and are available for consumption as they happen in near real time, represent the broadest cross-section of society at an international level, and are so inherently multifaceted. Tweets and Twitter's "following" mechanism link people in a variety of ways, ranging from short (but often meaningful) conversational dialogues to interest graphs that connect people and the things that they care about.

The potential of Twitter's self-organizing, ever-growing pool of data offers direct insight into trends and interests on both a personal and collective scale, but it has yet to fully capture the imagination of developers. And, in terms of the value that could come from data mining social media, Twitter is just the tip of the iceberg.

Monitoring this data effectively is challenging due to the quantity of data. In this project we proposed and developed a system that automatically summarizes people's opinions (twitter users) on features of various topics. The system is capable of streaming tweets, cleaning the noise, determine the features, and mine positive, negative, neutral opinions on the current topic's opinions. The system was built with a web interface to enable easy distribution of the service.

Other Achievements

- The system is capable of transforming a microblog into a formal English sentence in a process called data preprocessing.
- Using words like 'and' and 'or', the system was able to extract more product features using meronyms.

- Consideration of negation words (not, but, however, etc) when determining sentiment polarity improved the system's performance.
- A tweet sentiment classifier was built which was used to classify an entire tweet when the adjectives found were not found in the lexicon.
- Features were ranked before displaying them to the user.
- Feature pruning was performed on both term frequency, and meronym features. Redundant features were pruned.
- Fuzzy matching was performed in order to avoid detecting the same feature as two or more different ones. (E.g., i4n or iphone, should be detected as the same feature.)

Other than being a proof of concept, this project can be used by businesses, governments, or individuals to perform market research, check the public's reaction to policies, and find out what he world thinks about certain things respectively. The sentiment summarized report can be used by managers, or policy makers to influence decision making in organizations.

Challenges and Recommendations

- Insufficient Slang Dictionary – The slang dictionary used in this project was not representative of the entire slang used in micro blogs. The dictionary does not take context into consideration, as slangs might mean different things in different contexts.
- Lexicon – The lexicon does not contain all words that express sentiment, thus, a method was devised to expand the lexicon. However, the method used was also prone to error and might cause the system to become less accurate with time.
- Universal Knowledge base – In order for the system to achieve even higher accuracy, it is essential that is aware of the context/domain. Thus, having knowledge of the world might prove useful. The knowledge would help in the feature extraction process, especially in detecting related products in tweets.
- Opinion Mining – The opinion mining algorithm compares adjectives found within three words of a feature against the lexicon in order to determine the sentiment of that feature. This method has one major weakness, it assumes that any adjective close to a noun modifies that noun; this however, is not always the case. Devising a method to determine relationships between sentiment expressing words and features could greatly improve the accuracy of the system.

- Facebook – When we set out to do this project, we intended to mine both Twitter and Facebook. However, mining Facebook proved to be futile due to its restrictions imposed on the access token.
- Feature Detection Algorithm – The project employed the Apriori algorithm to detect features in a dataset. The Apriori has exponential run time, using the MaxMiner (linear order) algorithm instead could significantly reduce the run time of the extraction process.
- Sarcasm - a positive or negative sentiment word can switch sentiment if there is sarcasm in the sentence (e.g. "Sure, I'm happy for my browser to crash right in the middle of my coursework").

What Does The Future Hold?

Due to its large undiscovered territory, there are so many questions about social media analysis that remain unanswered.

For instance, can social sentiment media help us predict events? We've already seen a few attempts at this, as it happened with the election of Barack Obama against Mitt Romney. A more recent example comes from the UK, with an attempt from Tweview to predict the winner of the 2013 X Factor using sentiment, volume, and a lot of other social factors using the ever useful API from Brandwatch). While they predicted Nicholas McDonald as the winner of 2013 X Factor, he ended up second, with Sam Bailey winning

Could we perhaps adapt social predictions from social media for other scenarios, such as a viral campaign, or a PR disaster, or a business restructure or an application update?

8.0 REFERENCES

[1] Sundblad, H., "Automatic Acquisition of Hyponyms and Meronyms from Question Corpora", Proc. to the ECAI workshop on Machine Learning and Natural Language Processing for Ontology Engineering, Lyon, France, 2002.

[2] Hu, M. and Liu, B., "Mining Opinion Features in Customer Reviews", Proc. to 19th AAAI, San Jose, CA, USA, 2004, Pp. 755-760.

[3] Hearst, M., "Direction-Based Text Interpretation as an Information Access Refinement", Text-Based Intelligent Systems: current research and practice in information extraction and retrieval, Lawrence Erlbaum Associates, 1992.

[4] O'Connor, B., Balasubramanyan, R., Routledge, B. and Smith, N.A., "From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series", Proc. to International AAAI Conference on Weblogs and Social Media, George Washington University, Washington, DC, USA, 2010, pp.122-129.

[5] Kanayama, H. and Nasukawa, T., "Fully Automatic Lexicon Expansion for Domain-Oriented Sentiment Analysis", Proc. to Conference on EMNLP, Sydney, Australia, July 2006, pp.355-363.

[6] Makkonen, J., Ahonen-Myka, H. and Salmenkivi, M., "Simple Semantics in Topic Detection and Tracking", Information Retrieval, vol. 7, September-December 2004, pp. 347-368.

[7] Tatemura, J., "Virtual Reviewers for Collaborative Exploration of Movie Reviews", Proc. to the 5th International Conference on Intelligent User Interfaces, New Orleans, LA, USA, 2000, pp. 272-275.

[8] Scaffidi, C., Bierhoff, K., Chang, E., Felker, M., Ng, H. and Jin, C., "Red Opal: Product-Feature Scoring from Reviews", Proc. to the 8th ACM Conference on EC, San Diego, CA, USA, 2007, pp. 182-191.

[9] Pang, B., and Lee, L. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval* 2(1-2):1–135.

[10] Esuli, A., and Sebastiani, F. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC.*

[11] Hatzivassiloglou, V., and McKeown, K. 1997. Predicting the semantic orientation of adjectives. In *Proc. of ACL.*

[12] Kim, S.-M., and Hovy, E. 2004. Determining the sentiment of opinions. In *Proceedings of Coling.*

[13] Barbosa, L., and Feng, J. 2010. Robust sentiment detection on twitter from biased and noisy data. In *Proc. of Coling.*

[14] Bifet, A., and Frank, E. 2010. Sentiment knowledge discovery in twitter streaming data. In *Proc. Of 13th International Conference on Discovery Science.*

[15] Davidov, D.; Tsur, O.; and Rappoport, A. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of Coling.*

[16] Jansen, B. J.; Zhang, M.; Sobel, K.; and Chowdury, A. 2009. Twitter power: Tweets as electronic word of mouth. *Journal of the American Society for Information Science and Technology* 60(11):2169-2188.

[17] Tumasjan, A.; Sprenger, T. O.; Sandner, P.; and Welpe, I. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of ICWSM.*

[18] Mwanza, S.: Research and Implementation of Social Media Marketing Insights Extractor (2013).

[19] Kevin, C., Vidyasagar, P., and Tharam, D., "Content Quality Assessment Related Frameworks for Social Media", Digital Ecosystems and Business Intelligence Institute, Curtin University of Technology, Perth, Australia

[20]. Agichtein, E., Castillo, C., Donato, D., Gionis, A., Mishne, G.: Finding High-Quality Content in Social Media. In: Proceedings of the International Conference on Web Search and Web Data Mining, pp. 183–194 (2008)

[21] Cappiello, C., Francalanci, C., Pernici, B.: Data Quality Assessment from the User's Perspective. In: Proceedings of the 2004 International Workshop on Information Quality in Information Systems, pp. 68–73 (2004)

[22] Chai, K.: Development and Evaluation of User Contribution Scoring Models for Webbased Discussion Forums. Curtin University of Technology's Information Systems Honours Thesis (2007)

[23] Chen, Z., Xu, Y.: User-Oriented Relevance Judgement: A Conceptual Model. In: Proceedings of the 38th Hawaii International Conference on System Sciences, Track 4, p. 101b (2005)

[24] Ni, X., Xue, G.R., Ling, X., Yu, Y., Yang, Q.: Exploring in the Weblog Space by Detecting Informative and Affective Articles. In: Proceedings of the 16th International World Wide Web Conference, pp. 281–290 (2007)

[25] 12. McSherry, F., Najork, M.: Computing information retrieval performance measures efficiently in the presence of tied scores. Lect. Notes Comput. Sci. 4956, 414–421 (2008)

[26] T. Gilb. Principles Of Software Engineering Management. Addison-Wesley, 1988.

[27] Functional requirement: http://www.wikipedia.org (retrieved on 23rd July, 2014)

[28] Pak, A., and Paroubek, P. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proc. of LREC*

APPENDIX A

The tweet ranking algorithm used in this project is as follows:

1. *Procedure* ***TweetRanking()***
2. *begin*
3. *follower-followee_ratio_total_count; //holds all the ratios*
4. *reposts_count; // holds all the tweets repost counts.*
5. *//phase 1*
6. *for each tweet in the dataset;*
7. *get followers count;*
8. *get followees count;*
9. *subtract followees count from followers count to get the follower-followee ratio;*
10. *get the absolute value of the ratio;*
11. *add the ratio to follower-followee_ratio_total_count;*
12. *add repost count of this tweet to reposts_count;*
13. *end for;*
14. *//phase 2*
15. *for each tweet in the dataset;*
16. $Rank = \frac{absolute\ follower\text{-}followee\ ratio}{follower\text{-}followee\_ratio\_total\_count} * \frac{70}{100} + \frac{reposts\_count}{reposts\_count} * \frac{30}{100}$
17. *end for;*

APPENDIX B

These are some of the word used in the lexicon. Table A-1 shows the positive words and Table A- 2 shows the negative words. This is just a segment of the positive words used, the rest can be found on the attached CD or downloaded from http://www.cs.uic.edu/~liub/FBS/sentimentanalysis.html

admiringly	adorable	adore	adored	adorer
adoring	adoringly	adroit	adroitly	adulate
adulatory	advanced	advantage	advantageous	advantageously
advantages	adventuresome	adventurous	advocate	advocated
advocates	affability	affable	affably	affectation
affection	affectionate	affinity	affirm	affirmation
affirmative	affluence	affluent	afford	affordable
affordably	affordable	agile	agilely	agility
agreeable	agreeableness	agreeably	all-around	alluring
alluringly	altruistic	altruistically	amaze	amazed
amazement	amazes	amazing	amazingly	ambitious
ambitiously	ameliorate	amenable	amenity	amiability
amiably	amiable	amicability	amicable	amicably
amity	ample	amply	amuse	amusing
amusingly	angel	angelic	apotheosis	appeal
appealing	applaud	appreciable	appreciate	appreciated
appreciates	appreciative	appreciatively	appropriate	approval
approve	ardent	ardently	ardor	articulate
aspiration	aspirations	aspire	assurance	assurances
assure	assuredly	assuring	astonish	astonished

Table A- 1: Positive words used in the lexicon

abnormal	abolish	abominable	abominably	abominate
abomination	abort	aborted	aborts	abrade
abrasive	abrupt	abruptly	abscond	absence
absent-minded	absentee	absurd	absurdity	absurdly
absurdness	abuse	abused	abuses	abusive
abysmal	abysmally	abyss	accidental	accost
accursed	accusation	accusations	accuse	accuses
accusing	accusingly	acerbate	acerbic	acerbically
ache	ached	aches	achy	aching
acrid	acridly	acridness	acrimonious	acrimoniously
acrimony	adamant	adamantly	addict	addicted
addicting	addicts	admonish	admonisher	admonishingly
admonishment	admonition	adulterate	adulterated	adulteration
adulterer	adversarial	adversary	adverse	adversity
afflict	affliction	afflictive	affront	afraid
aggravate	aggravating	aggravation	aggression	aggressive
aggressiveness	aggressor	aggrieve	aggrieved	aggravation
aghast	agonies	agonize	agonizing	agonizingly
agony	aground	ail	ailing	ailment
aimless	alarm	alarmed	alarming	alarmingly
alienate	alienated	alienation	allegation	allegations
allege	allergic	allergies	allergy	aloof
altercation	ambiguity	ambiguous	ambivalence	ambivalent
ambush	amiss	amputate	anarchism	anarchism
anarchist	anarchistic	anarchy	anemic	anger
angrily	angriness	angry	anguish	animosity
annihilate	annihilation	annoy	annoyance	annoyances
annoyed	annoying	annoyingly	annoys	anomalous
anomaly	antagonism	antagonist	antagonistic	antagonize
anti-	anti-american	anti-israeli	anti-us	anti-white
antipathy	mess	messed	mire	misbehave

Table A- 2: Negative words used in the lexicon