UNIVERSITY OF NAIROBI
SCHOOL OF COMPUTING AND INFORMATICS

SOCIAL MEDIA SENTIMENT ANALYSIS FOR LOCAL KENYAN PRODUCTS AND SERVICES

BY

EDWIN WANYAMA NGERO
P58/72924/2009

JULY 2012

Submitted in partial fulfillment of the requirements of the degree of Master of Science in Computer Science

DECLARATION

This project, as presented in this report, is my original work and has not been presented for any other university award.

Edwin Wanyama Ngero

P58/72924/2009

Signed: ....[illegible]............

Date 31/Jul/2012

This project has been submitted as part fulfillment of the requirement for the award of Masters of Science in Computer Science of the School of Computing and Informatics of the University of Nairobi with my approval as the University Supervisor

Prof. Peter W. Wagacha

Signed: ...[illegible]...

Date: ....31st July 2012.

Dedication

I dedicate this work to the memory of my lovely parents who inspired, encouraged and provided for my education. This dedication also extends to my wife for the support during my project work.

ACKNOWLEDGEMENT

I take this opportunity to thank the School of Computing and Informatics, University of Nairobi for giving me a chance to pursue and successfully complete this course. My sincere thanks go to my supervisor, Prof. Peter Wagacha and the other members of staff in their various capacities for the untiring support, guidance and concern throughout my project work. I also thank my course mates for the encouragements and team work we have shared during this project. Above all, great thanks go to God for His providential care all through the time of my study.

Abstract

It has become a common practice on the web for a consumer to learn how others like or dislike a product before buying, or for a manufacturer or service provider to keep track of customer opinions on its products so as to improve the user satisfaction. However, as the number of reviews available for any given product or service grows, it becomes harder and more difficult for people to understand and evaluate what the majority opinions about the product are.

In this work, we attempt to resolve this problem for customer care service center in a Kenyan context by applying Sentiment analysis on people's opinions expressed on social media. Sentiment Analysis or opinion mining attempts to resolve this problem by first presenting the user with an aggregate view of the entire data set, summarized by a label or a score, and secondly by segmenting the opinions/sentiments into three classes (positive, negative and neutral) that can be further explored as desired.

We began by developing a module for facilitating searching, extraction and storage of opinions from social media. This was followed by development and training of a polarity classifier using the Python Natural Language Toolkit (NLTK) where we used the Naïve Bayes machine learning technique. The study went ahead to develop a web based application that integrates Facebook Graph API, Twitter API and the developed classifier to provide functionality for extracting, classifying and presenting classification results of data obtained from social media in a manner that can give a user with summarized as well as detailed view of customer opinion about a service or product.

We are able to achieve a relatively good accuracy of 78.9% with a modest dataset.

Table of Contents

List of Tables

List of Figures

Definitions of terms

Your **Wall** is where you and your friends can write on your Profile. Your friends may write on your Wall to communicate with you, congratulate you, embarrass you, and more. You post on your own Wall to let your friends know what you're up to.

The **News Feed** is a continuous stream of updates about your friends' activities on and off Facebook. It appears on your Home page.

A **post** is a message posted on ones wall. It's also called a status update. One may specify it as publicly accessible or restrict its accessibility by specifying it as private. Facebook provides a maximum length of 60,000 characters for a status update.

A **Friend** is someone you are connected to on Facebook.
Friending is the act of sending someone a friend request. All friendships have to be confirmed by both people in order for it to be official on Facebook.

Your **Profile** is your page. It contains your Wall, your photos and videos, a list of your friends, your favorite activities and interests, and anything else you choose to share.

A **Poke** is a casual gesture that means, "I'm thinking of you." The person you poke receives the Poke on her Home page when she logs in and has the choice to poke you back. The Poke is only visible to the person Poked.
Corpus is a large and structured collection of texts or writings that are usually electronically stored and processed.
Social Network is a social structure made up of persons called "nodes", which are tied or connected by one or more specific types of interdependency, such as friendship, kinship, common interest, financial exchange, dislike, sexual relationships, or relationships of beliefs, knowledge or prestige.
A tweet is an update message posted on ones profile on twitter. A tweet is limited to 140 characters.

CHAPTER 1

Introduction

1.1 Background to the study

The development and use of social network sites such as Facebook, twitter and Linked In have revolutionized the way most people in the current world share information and keep in touch with friends. People are able to express their opinions in form of posts, emoticons with regard to many issues that affect their day to day lives. In the recent past, people were restricted to use of emails and phones in communication as faster and convenient means of communication. These means are still in use and convenient but lack certain capabilities that are currently the selling features of social networks. For instance, with emails and phones one would be restricted to communicating to only those people whose contacts are known and with minimal or no information relating to friends of the party being communicated to. Social network sites provide space and mechanisms of making new friends and sharing information easily.

Due to the success of online social networking and media sharing sites and the consequent availability of a wealth of social data, social network analysis has gained significant attention in recent years. In spite of the growing interest, however, there is little understanding of the potential business applications of mining social networks (Bonchi, et al., 2011).

Despite being rich in content, social media unlike traditional media, have unorganized content contributed by users, often in fragmented and sparse fashion. Users wishing to get useful information usually have to spend a lot of their time filtering useless information and yet are not able to capture the essence. Though significant research efforts have been put in sentiment classification and analysis, most of the existing techniques rely on natural language processing tools to parse and analyze sentences in a review, yet they offer poor accuracy, because the writing in online reviews tends to be fragmented and less formal than writing in news or journal articles. Many opinion sentences contain grammatical errors and unknown terms that do not exist in dictionaries.

Sentiment analysis or opinion mining is a discipline closely related to data mining that uses machine learning techniques and natural language processing to determine what a certain group of people feels about an issue. Computers can use machine learning, statistic, and natural language processing techniques to perform automated sentiment analysis of digital texts on large collections of texts, including web pages, online news, internet discussion groups, online reviews, web blogs and social media.

1.2 Problem Statement

Social media today contains huge quantities of textual data, which are growing every day due to increased usage and entrants coupled with ease of generation of text and publishing. What is hard nowadays is not availability of useful information but rather extracting it in the proper context from the vast quantities of content (Yessenov and Misailovic, 2009). This information can provide some value to marketers or researchers on the general perception of their products or services. It is now beyond human power and time to seed through it manually therefore, the research problem of automatic categorization and organizing data is apparent.

It has been observed that whereas there is a large body of research on different problems and methods for social network mining, there is still a gap between the techniques developed by the research community and their deployment in real world applications (Bonchi, et al., 2011). Therefore the potential business impact of these techniques is still largely unexplored. A constant flow of information is generated as customers interact with a company. Harnessing the flow of information on social media and using it to bring the company ever closer to its customers is one of the applications that can benefit from sentiment analysis. By mining opinions to understand the nature of customer's issues and then proactively solving them can lead to money savings and improved customer satisfaction.

Indeed, Hu and Liu (2004) observe that in order to enhance customer satisfaction and shopping experience, it has become a common practice for online merchants to enable their customers to review or to express opinions on the products that they have purchased. With more and more common users becoming comfortable with the Web, an increasing number of people are writing reviews. As a result, the number of reviews that a product receives grows rapidly. Some popular products can get hundreds of reviews at some large merchant sites. Furthermore, many reviews are long and have only a few sentences containing opinions on the product. This makes it hard for a potential customer to read them to make an informed decision on whether to purchase the product. If he/she only reads a few reviews, he/she may get a biased view. The large number of reviews also makes it hard for product manufacturers and service providers to keep track of customer opinions of their products. For a product manufacturer, there are additional difficulties because many merchant sites may sell its products, and the manufacturer may (almost always) produce many kinds of products.

In this research we study the problem of classifying sentiments into positive, negative or neutral and generating summaries and trends of the classified data. These summaries are intended to find applicability in supporting customer care service centers in maintaining better customer relations. To tackle this problem, the study investigates issues that surround the development of a semantic analyzer of multilingual capability with the view of having it embedded in a web based application for use in rating opinions on products and services.

By applying sentiment analysis on product or service data obtained from Social Media, Customer Care Centers are able to address issues before they go viral to the extend escalating into product recalls or bad publicity for a company.

1.3 Objectives

The purpose of this project is to:-

1. Develop techniques for harvesting and storing data from Social Media
2. Using the techniques developed, harness and store data on various products and services in Kenya
3. Develop a Sentiment classifier that is able to classify sentiments in the stored Social Media data into positive, negative or neutral
4. Embed the classifier developed in a web based application for use in performing Social Media sentiment analysis on local Kenyan products and services.

1.4 Research Questions

1. What are the issues involved in harvesting data from Social Media? What is needed in harvesting data from Social Media on various products and services offered in Kenya? Are there suitable cost effective techniques that can be used for this task?
2. Which attributes of Social Media data can we obtain that are relevant for our study?
3. Which classification models have been successfully used before in similar research tasks? What are some of the issues that surround the usage of these models that would inform our selection of the most suitable model for use in carrying out this research work?
4. What is the best way to integrate the techniques learned and developed into an application that can provide Social Media sentiment analysis from the point of data collection to presentation of results? Which development platform renders itself suitable for integration of some of the heterogeneous tools that might be used?

1.5 Research outcomes and their significance to key audiences

The study is significant because it will involve work that will lead to:-

1. A sentiment analyzer algorithm that can be used on Kenyan issues.
2. A web based application that does sentiment analysis for Kenyan issues.
3. Providing various players Kenyan economy with ratings as regards consumer perceptions on services and goods given and other issues of concern that should be addressed for purposes of improvement.

1.6 Assumptions/Limitations of the Research

1. The words used on Facebook posts and Twitter do not fully constitute formal language. They involve acronyms, emoticons, slang and sheng.
2. Due to the limitation of time for this study only small sizes of the corpora will be used in the experiments.
3. Most people in Kenyan use more than two languages to express their feelings. We will restrict ourselves to text in English, Kiswahili and sheng.
4. We explore only comments on products and services in Kenyan.

CHAPTER 2

2.0 Literature Review

2.1 Social Media

Social media is an umbrella term that describes websites that connect individuals somehow. A hallmark of social media is the user generated content. This model contrasts with the editorially controlled style of old media. Social media is sometimes called Web 2.0.

The best way to define social media is to break it down. Media is an instrument on communication, like a newspaper or a radio, so social media is a social instrument of communication.

In Web 2.0 terms, this would be a website that does not just give you information, but interacts with you while giving you that information. This interaction can be as simple as asking for your comments or letting you vote on an article, or it can be as complex as the process of recommending movies to a user based on the ratings of other people with similar interests.

Regular media is synonymous to a one-way street where you can read a newspaper or listen to a report on television, but you have very limited ability to give your thoughts on the matter. Social media, on the other hand, is a two-way street that gives you the ability to communicate too.

Web 2.0 is a category of new Internet tools and technologies created around the idea that the people who consume media, access the Internet, and use the Web should not passively absorb what is available; rather, they should be active contributors, helping customize media and technology for their own purposes, as well as those of their communities. These new tools include, but are by no means limited to, blogs, social networking applications, RSS, social networking tools, and Wikis.

2.1.1 Examples of Social Media Websites

These include but are not limited to the following:-

Social Bookmarking: - These include Delicious *http://del.icio.us* and Blinklist *http://www.blinklist.com/.* Users interact by tagging websites and searching through websites bookmarked by other people.

Social News: - Sites under this category allow a user to interact by voting for articles and commenting on them. Examples under this category include *http://reddit.com* and *http://www.digg.com.*

Social Networking: - Social networking allows an individual to create a profile for themselves on the service and share that profile with other users with similar interests to create a social network. Users can choose to have public profiles which can be viewed by anyone or private profiles which can only be viewed by people that the users allow. Users can usually post photographs, music and videos on their site. Some of the most popular of these sites include Facebook *http://www.Facebook.com* and Twitter *http://www.twitter.com.*

Social Photo and Video Sharing: - These sites are also known as Content Hosting Services. Content hosting or content sharing sites allow users to upload content that they have created for others to view. Two of the most popular of these sites are YouTube www.youtube.com for videos and Flickr *www.flickr.com* for photographs. Users can also create an individual profile and list their favorite photos or videos. Users are able to rate and comment on the videos or photos posted and provide feedback to the creator and other users.

Wikis: - This is a collaborative website that anyone within the community of users can contribute to or edit. A wiki can be open to a global audience or can be restricted to a select network or community. Wikis can cover a specific topic or subject area. Wikis also make it easy to search or browse for information. Although primarily text, wikis can also include images, sound recordings & films. Wikipedia *http://en.wikipedia.org* the free internet encyclopedia is the most well known wiki.

2.1.2 Growth of Social Media

According to Corbo (2012), the 1990s ushered in a number of innovations that were exciting to Internet communities at the time. The earliest versions of web pages and webcams emerged in 1991, as did the adoption of mp3 as a standard audio file. Geocities surfaced in 1997, providing a platform for individuals and businesses to operate their own websites within Geocities' virtual communities. Also debuting in 1997 were AOL's Instant Messaging and Sixdegrees, a social networking service that introduced the concepts of friends in social networking communities. This era's advances would prove to be foundational to social media platforms and activities down the road.

The 2000s brought a tremendous growth in social networking platforms. Friendster opened its virtual doors in 2002, followed quickly in 2003 by Myspace, originally a Friendster clone. 2003 also brought the world Skype and Linkedin, representing the vast potential of VoIP and business networking respectively.
The opening of Facebook in year 2004 represented one of the most well-known events in social media history. It was the year the term social media was accepted into mainstream consciousness. Twitter, was born in 2006 and is credited with facilitating monumental social change across the world. 2007 gave the world the iPhone, which is also now etched into cultural consciousness. It launched a fervent and more lasting interest in mobile Internet activities, a great many of them with a social element. New options for Internet-based social interactions continue to arise, the recent launch of Google+ being one of them.

Analytics have changed over time to keep pace with the innovations in online community platforms. Marketing analysis has evolved to take advantage of the constantly growing reach of social media. Website statistics have expanded from simple analysis of Internet user behavior to contemporary analysis that includes social analytics. By 2010, even URL shortening services such as bit.ly were providing analytics, to help individuals and businesses track link distribution.

Over the years, people have demonstrated a strong desire to utilize the Internet to connect with others, to be social and find others with whom to relate and share interests. Businesses have studied these wishes and adapted to them, to tap into social networking tendencies as a way to find and cultivate customers, just as customers find and cultivate relationships amongst one another and also to get feedback from customers over services and goods offered. Whatever the future of Internet-based communication holds, it is assured that social media elements will remain significant.

2.1.3 Users of Social Media

According to Openbook (2011), a research done in 2011 found out that Facebook, Youtube and Twitter form the cream of the most popular top 10 social networking sites used in Kenya.

Similar research available on Socialbakers (2012) indicates that Facebook penetration in Kenya in April 2012 was at 3.28% compared to the country's population and 32.87% in relation to number of Internet users. The total number of FB users in Kenya had surpassed 1313400 having grown by more than 77600 users in a period of 6 months preceding April 2012. The user age distribution on Facebook in Kenya indicates that the largest age group is currently 18 - 24 with total of 512 226 users, followed by the users in the age of 25 – 34 as shown in the Figure 1.

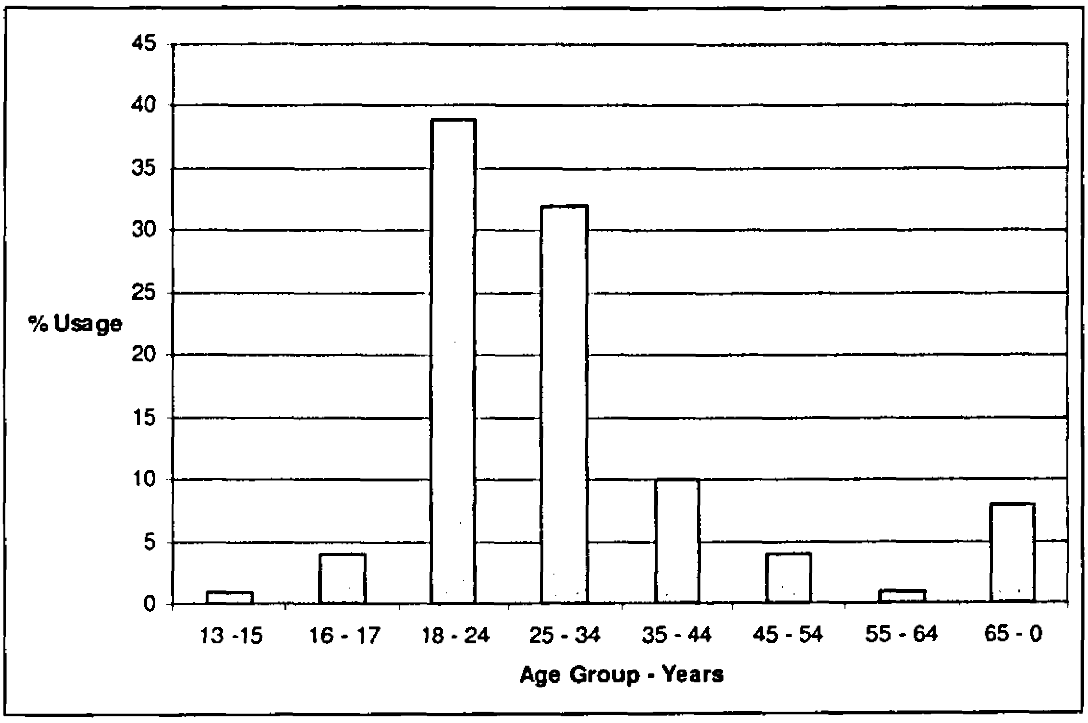


Figure 1. User age Distribution on Facebook in Kenya (Source - Socialbakers 2012)

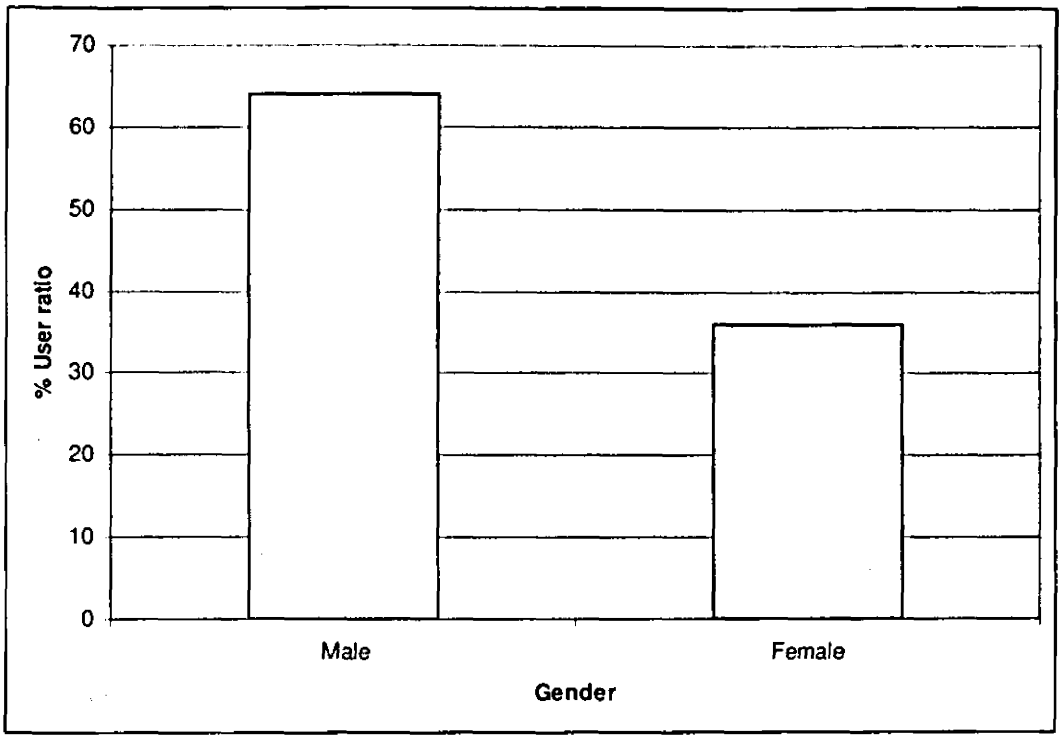


Figure 2. Male/Female User Ratio on Facebook in Kenya (Source - Socialbakers 2012)

The statistics also show that there are 64% male users and 36% female users in Kenya as at April 2012 shown in Figure 2.

2.1.4 Setup of Social Media

The current successes attained by social media would not have been possible without Web 2.0 technologies. As mentioned earlier in section 2.1, Web 2.0 is a category of new Internet tools and technologies created around the idea that the people who consume media, access the Internet, and use the Web shouldn't passively absorb what's available; rather, they should be active contributors, helping customize media and technology for their own purposes, as well as those of their communities. In order to promote the harnessing of collective knowledge from people, websites must be easy enough to use that they don't stand in the way of people using the Internet to share their knowledge.

Thus, whereas Web 2.0 is about creating a social web, it is also about creating a more interactive and responsive web. It is in this way that technologies such as AJAX, RSS and

Eclipse become central to the idea of Web 2.0. It is worth mentioning that though many social media sites were built and run on different technologies, they all provide for these technologies to enrich user experience.

AJAX, which stands for Asynchronous JavaScript And XML, allows websites to communicate with the browser behind the scenes and without human interaction. This means you don't have to click on something for the web page to do something.

RSS which stands for RDF Site Summary, is an XML-based vocabulary for distributing Web content in opt-in feeds. Feeds allow the user to have new content delivered to a computer or mobile device as soon as it is published. An RSS aggregator or RSS reader allows the user to see summaries of all their feeds in one place. Instead of visiting multiple Web pages to check for new content, the user can look at the summaries and choose which sites to visit for the full versions.

Eclipse is a Java-based open source platform that allows a software developer to create a customized development environment (IDE) from plug-in components built by Eclipse members. Eclipse is managed and directed by the Eclipse.org Consortium. The major advantage offered by Eclipse development platform is that it allows an IT department to mix and match development tools rather than being committed to a single vendor's suite of development products. Though written in Java, the Eclipse Platform supports plug-ins that allow developers to develop and test code written in other languages. Such capabilities have offered freedom and flexibilities that promote innovation for development of interfaces that afford better user experiences.

2.2 Concept of Sentiment Analysis

When conducting research or making every day decisions, we often look for other people's opinions. We consult political discussion forums when casting a political vote, read consumer reports when buying appliances, ask friends to recommend a restaurant for the evening. And now Internet has made it possible to find out the opinions of millions of people on everything from latest gadgets to political philosophies (Mejova, 2009).

The internet today contains huge quantities of data that keeps growing day by day. A large number of websites have come up that provide people with an opportunity to express their opinion about certain things. These include product review sites, forums, blogs and social network sites. Consequently as a response to the growing availability of informal, opinionated texts like blog posts and product review websites, a field of Sentiment Analysis has sprung up in the past decade to find out what people feel about a certain issue.

According to socialtimes (2011), researchers at eBay research labs expressed that properly conducted sentiment analysis provides more usable information than surveys and focus groups, where participants may want to complete the process quickly or express less than honest opinions due to the influence of others.

Sentiment Analysis is an active area of ongoing research. It is a field of study that aims to uncover the attitude of the author on a particular topic from the written text.

2.2.1 Examples of Sentiment Research

Sentiment analysis has been handled as a Natural Language Processing task at many levels of granularity. Starting from being a document level classification task (Pang and Lee, 2004), it has been handled at the sentence level (Hu and Liu, 2004) and more recently at the phrase level (Wilson et al., 2005; Agarwal et al., 2009).

Most research work in Sentiment analysis work has been done on product and movie reviews, where it is easy to identify the topic of the text. This is because of the ready availability of product review datasets (Mejova, 2009). Pang et al. (2002) conducted an extensive experiment on movie reviews using three traditional supervised machine learning methods (i.e., Naive Bayes (NB), maximum entropy classification (ME), and support vector machines (SVM)). His results indicate that standard machine learning techniques definitively outperform human produced baselines. However, he found that machine learning methods could not perform as well on sentiment classification as on traditional topic based categorization. Their results indicated that use of machine learning methods is more superior to the use of human generated baselines. The advantage is that reviews from such sites already have a clearly specified topic which is often assumed that the sentiments expressed in the reviews have to do with the topic.

Yessenov and Misailovic (2009) also carried out sentiment analysis on movie review data comments from the popular social network Digg as their data set and classified text by subjectivity/objectivity and negative/positive attitude. They used different approaches in extracting text features such as bag-of-words model, use of large movie reviews corpus, restricting to adjectives and adverbs, handling negations, bounding word frequencies by a threshold, and using WordNet synonyms knowledge. They then evaluated their effect on accuracy of four machine learning methods - Naive Bayes, Decision Trees, Maximum-Entropy, and K-Means clustering. Their results showed that simple bag-of-words model performed relatively well and could be further refined by the choice of features based on syntactic and semantic information from the text.

Pak et al (2009) used emoticons to identify tweets that were to be used to train the classifier. Their tests were able to classify texts as positive, negative or neutral. However, the only language they supported in their text was English.

Abbasi et al (2007) explored the use of sentiment analysis methodologies in the classification of web forum opinions written in multiple languages. They evaluated the utility of stylistic and syntactic features in sentiment classification of English and Arabic content on movie review dataset, U.S and Middle Eastern web forum postings. Specific feature extraction components were integrated to account for the linguistic characteristics of Arabic. The Entropy Weighted Genetic Algorithm (EWGA) was also developed, which is a hybridized genetic algorithm that incorporates the information gain heuristic for feature selection. Their experiments indicated that stylistic features significantly enhanced performance across all test beds while EWGA also outperformed other feature selection methods, indicating the utility of these features and techniques for document level classification of sentiments.

Cui et al (2006) identified two drawbacks faced by most of the current researches in Sentiment Analysis. The first drawback concerns large scale real world data sets that current researches have to contend with. Most previous work was conducted on relatively small, experimental data sets employing at most a few thousand articles. Building sentiment classifiers for web-scale processing, with all the attendant problems of pre-processing, spelling/grammar mistakes, broken HTML, etc., brings in a whole set of other problems. Data sets are in the hundreds of thousands, sometimes millions, and are neither clean nor consistent. Algorithms need to be efficient and subtle language effects become visible. The second drawback has to do with the feature model employed i.e. Unigram versus N-gram. Current work mostly focuses on using unigrams and bigrams rather than higher order n-gram models to capture sentiments in the text. They considered that that those experiments were hindered by the small training corpora, and thus were not able to show the effectiveness of high order n-grams (n _ 3) in discerning subtleties in expressing sentiments.

To remedy these problems, they made three contributions in their research work. First, they conducted experiments on a corpus of over 200k online reviews with an average length of

over 800 bytes crawled from the Web. Such a large-scale data set allowed them not only to train language models and cull high order n-grams as features, but also to study the effectiveness and robustness of classifiers in a simulating context of the Web.
Secondly, they studied the impact of higher order n-grams (n _ 3). Compared to previous published work – they showed that there is a benefit to use higher order n-grams beyond unigrams and bi-grams. This was done alongside other experiments that employed other features beyond n-grams, such as POS tags and parse trees which were found to lack significant improvements.
They also studied multiple classification algorithms for processing large-scale data. In this regard three algorithms: (i) Winnow, (ii) a generative model based on language modeling, and (iii) a discriminative classifier that employs projection based learning were explored. Their experimental results showed that the discriminative model significantly outperformed the other two kinds of models.

Hu and Liu (2004) performed a research in which they mined and summarized customer reviews crawled from the web over a certain product. They studied the problem of producing feature based summaries of customer reviews of products sold online. Their task was performed in three steps that involved: (1) identifying features of the product that customers have expressed their opinions on (called product features); (2) for each feature, identifying review sentences that give positive or negative opinions; and (3)
producing a summary using the discovered information. A number of novel techniques such as part of speech tagging, frequent and infrequent features identification compactness pruning and redundancy pruning were used to achieve results.

Gitau and Miriti (2011) carried out sentiment analysis using unigrams, emoticons and bigrams that were mined from Twitter on Kenyan issues. They used the Naïve Bayes model to perform polarity classification of tweets into positive and negative. The results achieved from their work suggest that the Naïve Bayes Model of classification can be used as a starting point with relative ease to perform Sentiment Analysis with good results on Social Media. They suggest further work to be done on additional Social Media players such as Facebook and Youtube.

2.2.2 Elements used in Sentiment Analysis

Liu (2010) identifies that five elements are used in sentiment analysis. These include the object, attribute, opinion holder, opinion orientation and strength and time. An object refers to the target entity that has been commented on. It may have a set of components (or parts) and a set of attributes (or properties) which are collectively called the features of the object. We can use n example of a particular brand of cellular phone as an object. It has a set of components (e.g., battery and screen), and also a set of attributes (e.g., voice quality and size), which are all called features. An opinion can be expressed on any feature of the object and also on the object itself. For example, in "I like iPhone. It has a great touch screen", the first sentence expresses a positive opinion on "iPhone" itself, and the second sentence expresses a positive opinion on its "touch screen" feature. Opinion holder: The holder of an opinion is the person or organization that expresses the opinion.

An opinion holder is the person who expresses opinion about the object. For product reviews and blogs, opinion holders are usually the authors of the posts. Opinion holders are more important in news articles because they often explicitly state the person or organization that holds a particular opinion. An opinion on a feature f (or object o) is a positive or negative view or appraisal on f (or o) from an opinion holder. Positive and negative are called opinion orientations.

2.2.3 Feature based Sentiment Analysis Model

With the concepts in section 2.3 above in mind, Liu (2010) defines a model of an object, a model of an opinionated text, and the mining objective, which are collectively called the feature-based sentiment analysis model.

Model of an object: An object o is represented with a finite set of features, F = {f1, f2, ..., fn}, which includes the object itself as a special feature. Each feature fi of F can be expressed with any one of a finite set of words or phrases Wi ={wi1, wi2, ..., wim}, which are synonyms of the feature.

Model of an opinionated document: A opinionated document d contains opinions on a set of objects {o1, o2, ..., or} from a set of opinion holders {h1, h2, ..., hp}. The opinions on each object oj are expressed on a subset Fj of features of oj.

2.2.4 Types of opinions

An opinion can be either one of the following two types:-

Direct opinion: A direct opinion is a quintuple (oj, fjk, ooijkl, hi, tl), where oj is an object, fjk is a feature of the object oj, ooijkl is the orientation of the opinion on feature fjk of object oj, hi is the opinion holder and tl is the time when the opinion is expressed by hi. The opinion orientation ooijkl can be positive, negative or neutral.

Comparative opinion: A comparative opinion expresses a preference relation of two or more objects based on some of their shared features. It is usually conveyed using the comparative or superlative form of an adjective or adverb, e.g., "Coke tastes better than Pepsi".

2.3 Objective of Sentiment analysis

Liu (2010) sums up his observations that given an opinionated document d, the Objectives of sentiment analysis on direct opinions are:

1. To discover all opinion quintuples (oj, fjk, ooijkl, hi, tl) in d, and
2. To Identify all synonyms (Wjk) of each feature fjk in d.

Liu further observes that not all five pieces of information in the quintuple need to be discovered for every application because some of them may be known or not needed. For example, in the context of online forums, the time when a post is submitted and the opinion holder are all known as the site typically displays such information.

A Sentiment can be identified at several levels which include the overall document (e.g., product review, blog, forum post), a sentence or a specific object attribute. For each level, search and analysis operates under somewhat different assumptions.

2.4 Technical challenges

A number of technical challenges have been observed in sentiment analysis (Liu, 2010). The first is Object identification. A blog may have sentiments expressed on different objects. In such a case the problem lies in identifying the object on which a sentiment has been expressed without which the opinion is of little use. In a typical opinion mining application, the user wants to find opinions on some competing objects (e.g., products). The system thus needs to separate relevant objects and irrelevant objects.

The second is feature extraction and synonym grouping. Current research mainly finds nouns and noun phrases. Although the recall may be good, the precision can be low. Furthermore, verb features are common as well but harder to identify. To produce a good summary there is need to group synonym features as people often use different words or phrases to describe the same feature (e.g., "voice" and "sound" refer to the same feature). This problem is also very hard. A great deal of research is still needed.

The third challenge is on opinion orientation classification. The task here is determination of whether there is opinion on a feature in a sentence, and if so, whether it is positive or negative. Existing approaches are based on supervised and unsupervised methods. One of the key issues is to identify opinion words and phrases (e.g., good, bad, poor, great), which are instrumental to sentiment analysis. The problem is that there are seemingly an unlimited number of expressions that people use to express opinions, and in different domains they can be significantly different. Even in the same domain, the same word may indicate different opinions in different contexts. For example, in the sentence, "The battery life is long" "long" indicates a positive opinion on the "battery life" feature. However, in the sentence, "This camera takes a long time to focus", "long" indicates a negative opinion.

Fourthly integration of the five pieces of information in the quintuple so as to get a match is a complex task. That is, an opinion must be given by an opinion holder on a certain feature of an object at a certain time. To make matters worse, a sentence may not explicitly mention some pieces of information, but they are implied due to pronouns, language conventions, and the context. To deal with these problems, we need to apply NLP techniques such as parsing, word sense disambiguation and coreference resolution in the opinion mining context. Other challenges include domain dependency, irony and sarcasm and social elements. Awareness of the multifaceted nature of opinions gives us a foundation upon which to conduct our search and analysis.

2.5 Privacy Concerns

In recent years, social network research has been carried out using data collected from online interactions and from explicit relationship links in online social network platforms such as Facebook, LinkedIn, Flickr and Instant Messenger (Bonchi, et al., 2011).

The mining of this data makes it difficult for an individual to autonomously control the unveiling and dissemination of data about his/her private life (Wel & Royakkers, 2004). Sentiment analysis or opinion mining involves the use of personal data of some kind and can lead to the disruption of some important normative values. One of the most obvious ethical objections lies in the possible violation of peoples' informational privacy. Protecting the privacy of users of the Internet is an important issue. Informational privacy mainly concerns the control of information about oneself. It refers to the ability of the individual to protect information about oneself. The privacy can be violated when information concerning an individual is obtained, used, or disseminated, especially if this occurs without their knowledge or consent. Wel and Royakkers note that the privacy issues due to web mining often fall within this category of informational privacy. They further state that the value of peoples' individualism is violated when they are judged and treated based on patterns resulting from web-data mining.

When data obtained from the web is made anonymous the discovered information no longer links to individual persons, and there is no direct sense of privacy violation because the data is not then directly linked to a person.

This study intends to use data from Facebook and Twitter that is specified as public by the users who generated it. The data will be made anonymous to protect the privacy of individuals by picking attributes of posts and tweets that exclude the author's identity. We will therefore collect public messages, the corresponding message identifiers and the time the messages were created for purposes of this study. Facebook clarifies on its privacy statement that information made public on its site can be viewed by anyone even off Facebook. Such information also shows up when someone does a search on Facebook or on a public search engine. The information can also be accessed by the games, applications, and websites one visits with friends and is also accessible to anyone who uses its APIs such as the Graph API.

2.6 Conceptual Model

Based on the insight obtained from the relevant literature cited above, this paper draws upon three approaches:-

i. Pattern based approach where we use the unigram model of features
ii. Simple Bag of words model with information gain heuristic
iii. Machine Learning, where we use the Naïve Bayes technique.

These three approaches inform the creation of a conceptual model shown in Figure that is used in this research work.

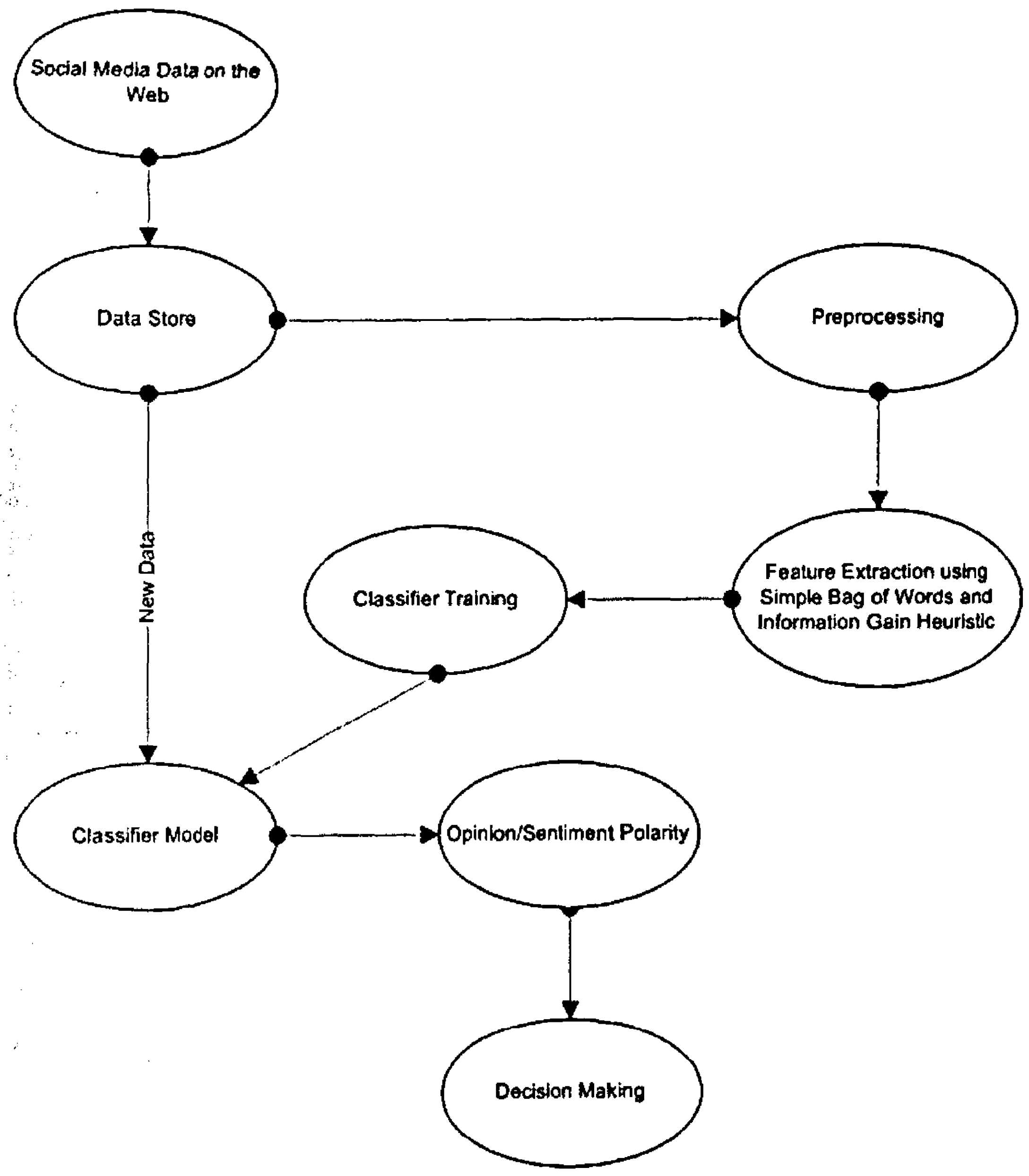

Social Media Data on the Web
Data Store
Preprocessing
New Data
Feature Extraction using Simple Bag of Words and Information Gain Heuristic
Classifier Training
Classifier Model
Opinion/Sentiment Polarity
Decision Making

Figure 3 Conceptual Model

CHAPTER 3

3.0 Methodology

This chapter presents the research methods that were followed to achieve the overall objectives outlined in section 1.3 above. We explain the sources of training data that were used, data cleaning, the way features were selected and categorized, how classification was achieved and finally how the classifier will be evaluated. The following activities were carried out in this research:

a) Data sources and collection mechanisms
b) Data cleaning and categorization into three categories:-
    i. Positive
    ii. Negative and
    iii. Neutral
c) Classifier development
d) System Design
e) System implementation
f) Run some test and Evaluate the results
g) Discuss results

A summary of the activities carried out to achieve objectives specified in section 1.3 above are as tabulated:-

No	Objective	How objective was achieved
1	Collect, clean and categorize classifier training data from Facebook	The Graph and Twitter APIs were customized and used to fetch data from Facebook and Twitter respectively. The fetched data was stored in a database then eventually cleaned and stored in files labeled positive, negative or neutral based on polarity of a tweet or message.
2	Develop a Sentiment classifier that is able to classify domain specific sentiments in posts as positive, negative or neutral	The study applied machine learning technique to develop and train a sentiment classifier based on the features extracted from data collected from Twitter and Facebook. Natural language toolkit of Python

		programming language was used in implementing the classifier.
3	Provide a tool that performs Sentiment Analysis for Kenyan issues based on the classifier developed.	We developed a web based application that provides interfaces for use in searching and fetching Facebook data. It then uses the classifier that was developed and trained to performing Sentiment Analysis on entities/topics in Kenya.

Table 1. Activities carried out to achieve objectives

3.1 Research design

This research design was chosen because of the sparse and informal nature of Social Media data, a scenario that necessitates use of machine learning and computational linguistics techniques. Traditional approaches of natural language processing are challenged by the lack of language formality usage on Social Media.

3.2 Data Collection Methods and Tools

The source of data for this study is Facebook and Twitter. This data includes text, emoticons and acronyms. Facebook and Twitter provide millions of users an opportunity to express their personal opinions about any topics. Thus the data available on this site and other social network sites has immediate applicability in business environments such as gaining information from summarized views of users on certain products or services (Yessenov and Misailovic, 2009).

Data for this study was collected from Facebook and Twitter using APIs that are readily available and free for use in accessing data from the two social networks. It is a fact that the style and nature of writing posts on Facebook and Twitter is not strictly formal. The data collected therefore had to go through some cleaning exercise before it was used in training the classifiers. We therefore had to remove irrelevant characters such as URL links and repeated characters. This was meant to ensure that the classifiers are trained on appropriate data for better performance.

3.2.1 Feature selection/extraction
In order to perform machine learning, it is necessary to extract clues from the text that may lead to correct classification (Yessenov and Misailovic, 2009). Clues about the original data are usually stored in the form of a feature vector, F = (f1; f2; : : : fn). Each coordinate of a feature vector represents one clue, also called a feature, fi of the original text. The value of the coordinate may be a binary value, indicating the presence or absence of the feature, an integer or decimal value, which may further express the intensity of the feature in the original text. In most machine learning approaches, features in a vector are considered statistically independent from each other.

The selection of features strongly influences the subsequent learning. The goal of selecting good features is to capture the desired properties of the original text in the numerical form. The study endeavored to select the properties of the original text that are relevant for the sentiment analysis task. Unfortunately, the exact algorithm for finding best features does not exist. We therefore relied on intuition, domain knowledge, and experimentation for choosing a good set of features.

3.2.2 Bag of words
We used a simple bag of words model to perform feature extraction. Usually opinion detection is based on the examination of adjectives in sentences though this can be misleading in cases that include sarcastic sentiments and negations. As a result we considered polarities based on sentences more than they appear based on adjectives.

3.2.3 Classification
This research used a classification algorithm to predict the label for a given input sentence. There are two main approaches for classification: supervised and unsupervised. In supervised classification, the classifier is trained on labeled examples that are similar to the test examples, whereas unsupervised learning techniques assign labels based only on internal differences (distances) between the data points. In this classification approach each sentence is considered independent from other sentences (Yessenov and Misailovic, 2009). The label we were interested in this project is the polarity of the sentence. We built a classifier based on the Naïve Bayes method which is a supervised classification technique, and trained it to

perform classification. A model of the supervised classifier used in this research is as shown in figure 3.

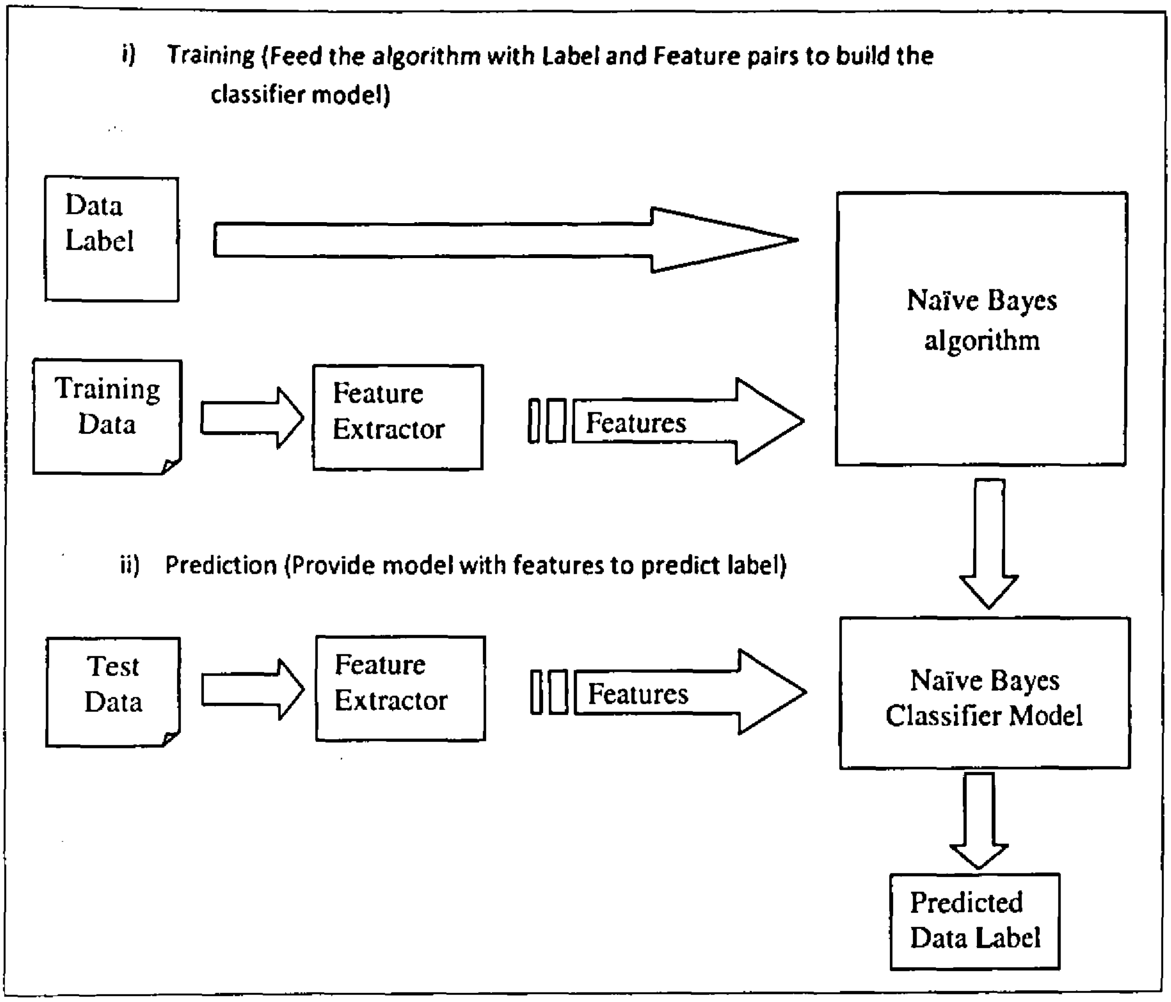

i) Training (Feed the algorithm with Label and Feature pairs to build the classifier model)
Data Label
Training Data
Feature Extractor
Features
Naïve Bayes algorithm
ii) Prediction (Provide model with features to predict label)
Test Data
Feature Extractor
Features
Naïve Bayes Classifier Model
Predicted Data Label

Figure 4. Supervised classifier Model

Naive Bayes assumes that all features in the feature vector are independent, and applies Bayes' rule on the sentence. Naive Bayes calculates the prior probability frequency for each label in the training set. Each label is given a likelihood estimate from the contributions of all features, and the sentence is assigned the label with highest likelihood estimate. The data corpus was divided into two groups – training set and test set. The training of the classifier was done on the sentences from the training set.

The task of building the classifier was carried out using Python as a programming language. Python's Natural Language Toolkit (NLTK) suite of libraries has rapidly emerged as one of the most efficient tools for Natural Language Processing. According to semanticbible (2008) Python has the following desirable features that made it suitable for use in achieving objectives of this study:-

i. It is a clean, elegant, object-oriented language
ii. It has a rich set of library modules
iii. There is an active and readily available open-source community that offers support
iv. It provides interactive evaluation that supports learning and rapid development
v. It is broadly used in scientific research

A similar view is held that NLTK was designed with four primary goals in mind (Bird, et al., 2009):

Simplicity

To provide an intuitive framework along with substantial building blocks, giving users a practical knowledge of NLP without getting bogged down in the tedious house-keeping usually associated with processing annotated language data

Consistency

To provide a uniform framework with consistent interfaces and data structures, and easily guessable method names

Extensibility

To provide a structure into which new software modules can be easily accommodated, including alternative implementations and competing approaches to the same task

Modularity

To provide components that can be used independently without needing to understand the rest of the toolkit

3.3 Implementation of the Sentiment Analysis Tool in a web based application

The trained and tested classifier was used to develop a sentiment analyzer that was implemented in a web based application for use in Kenya. The application is intended to provide ratings on various topics.

3.4 Evaluation

The following criteria were used to evaluate the overall system upon completion of development. A positive answer to each of the questions below confirmed the success of this research.

i. Is the tool able to collect training data from Facebook in both English and Swahili?
ii. Is the chosen method of classification ideal for the kind of data that has been collected?
iii. Once the data is collected, is the classifier able to be trained?
iv. Are we using the correct features for classification?
v. Are the results of the tests on the data accurate in terms of quality, or are the results unfavorably skewed towards one sentiment?
vi. Is the Web application developed able to provide sentiment analysis on data of topical issues extracted from Facebook and Twitter?

In the evaluation of the classifier we used Natural Language Toolkit inbuilt metrics to measure accuracy, precision and recall of the developed classifier.

3.5 Execution of methodology

3.5.1 API customization for interfacing with Social Media

Facebook and Twitter both provide APIs that can be used for searching and fetching post messages and tweets that have keywords specified in the search criteria. With these tools one is able to fetch posts and tweets on topics of interest from the two social networking sites. Search mechanisms and requirements for the two sites differ as explained below:-

Facebook data access:- Facebook provides search capability through the Graph API. The Graph API as such allows you to easily access all public information about an object. You can search over all public objects in the social graph with https://graph.facebook.com/search. We used https://graph.facebook.com/search? fields=message&q=query &type=post&

access_token=value. The message part specifies that we only want to view the message attribute of the results, query specifies the item we are searching for while type specifies that we are searching for posts in the social network. The access token value is meant to fulfill the requirement for extended permissions that enable the search to read from the message stream. We obtained an access token form the Graph API explorer of Facebook. To obtain an access token one is required to sign in to Facebook, navigate to the Graph API explorer app, select the type of permission required then generate the token using the interface provided.

Twitter data access:- Twitter provides search capability through *http://search.twitter.com/search.json?q='. $searchfor.* The json bit tells twitter to return data in json format while the q parameter specifies the item we are searching for.

3.5.2 Data Collection

We used keywords for current topical issues as search criteria for collecting relevant data from Facebook and Twitter. This exercise was repeated a number of times with the result being a dataset of over 3000 posts fetched so far from both Facebook and Twitter on various topical issues.

The data fetched from the two sites was then decoded from JSON (JavaScript Object notation) to text and stored in a MySQL database. We only stored Facebook posts whose message does not exceed 200 characters (Facebook allows message sizes of up to 60000 characters). All tweets fetched were stored since they don't exceed 140 characters. For each post fetched we stored the following attributes:-

i. The unique post identifier
ii. Post message
iii. The time the post message was created.
iv. The item searched for
v. The source of the message i.e. whether Facebook or Twitter.
vi. Polarity which is null as at the point of fetching the data. This field is later updated during classification of the message's polarity.

3.5.3 Data Preparation for Classifier training

The raw data harnessed from Facebook and Twitter is directly unsuitable for use in training a polarity classifier since it often has unwanted characters such as links, exclamation marks question marks and other irrelevant characters. These characters are not of essence to this study except for emoticons that are used as means of expression.

The entire corpus stored in the MySQL database was then picked and transferred it to three text files. The three files correspond to the three categories of features on which the classifier is to be trained namely positive, negative and neutral. Time was taken to manually read and clean the contents of each file line by line while removing irrelevant characters, words and sentiments. The result of this exercise was three files each of which is a collection of sentiments that express one category of polarity. We currently have 981 negative sentiments in the file for negatives, 367 neutral sentiments in the file for neutrals and 559 positive sentiments in the file for positives.

During training of the classifier, 75% of the cleaned data combined is used as training data while 25% of the data is used as the testing set.

3.5.4 Naïve Bayes Classifier

The Naïve Bayes Classifier is based around the Bayes rule which is a way of looking at conditional probabilities that allows you to flip the condition around in a convenient way. A conditional probability is a probability that event X will occur, given the evidence Y. That is normally written P(X | Y). The Bayes rule allows us to determine this probability when all we have is the probability of the opposite result, and of the two components individually:-

$$P(X \mid Y) = P(X)P(Y \mid X) / P(Y).$$

3.5.5 Feature Extraction and Classifier Development

The NLTK toolkit was used to realize a feature extractor and a Naïve Bayes classifier. The Naïve bayes classifier requires that the training features be rendered in 'feature label' pair for it to recognize that a particular feature belongs to a particular label. In our case the file category (negative, positive or neutral) from which a sentiment comes from is the label for that sentiment while the feature is the sentiment. Since our sentiment classification is restricted to the sentence level, the data had to be split into a file per sentiment per category. We use scripts to achieve this splitting. We then use an algorithm in the classifier program that returns a dictionary of feature sets that are rendered as 'feature label' pair as explained above. The returned dictionary of feature sets enables the classifier to learn to associate a feature with a particular label that is positive, negative or neutral. The algorithm is also structured to discard stop words from the dictionary of feature sets used for training.

We used the Naive Bayes classifier that ships with the Natural Language toolkit and customized it to suit our research needs. Its customization is currently complete and functional. It does the following:-

1. Makes the necessary imports of resources such as the categorized corpus reader that we use to read training data stored in files and a set of stop words.
2. Reads in training data from files.
3. Obtains the categories of labels to be used.
4. Generates a feature set for training the classifier using a bag of words model.
5. Splits the feature set into a training set and testing set.
6. Trains a Naïve Bayes classifier imported from NLTK toolkit using the training set.
7. Tests the classifier accuracy using the testing set.
8. Prints out the accuracy and informative features.
9. Retrains the classifier with the entire (combined Training and testing set) set.
10. Connects to the database and loops through each message or tweet as it classifies and updates the polarity of each record.

The classifier has been tested and found to be working at an accuracy of 78.9% according to its inbuilt NLTK accuracy module. A look at the classified records shows that the classifier is able to determine the polarity of most of the sentiments.

3.5.6 Evaluation of Classifier

In experimenting with the Naïve Bayes Classifier, we relied on the NLTK metrics module which provides functions for calculating accuracy, precision and recall for the Classifier

3.5.7 Implementation of the system

The Sentiment Analyzer System was implemented using Java language for the application logic whereas the interface was realized using Java Server Faces (JSF). MySQL database was used as the back end for storing data fetched from Facebook and Twitter and other application data.

CHAPTER 4

4.0 Design and Implementation of Sentiment Analyzer System

4.1 System Design Specification

In this section we specify the functionalities of the system, associated inputs and outputs together with associated data sources. We employed the use of agile software development Methodology to guide in the object oriented development of the application software.

4.1.1 System Description

The Sentiment Analyzer System was designed to carry out Sentiment Analysis for Kenyan issues based on data obtained from Facebook and Twitter. This is inline with our earlier stated objectives in section 1.3 above. To ensure realization of the objectives, the research designed the application with four main modules outlined below:-

i. Facebook and Twitter API interface modules
ii. Classification Module
iii. Reports Module
iv. User administration

The research employed Use Cases to describe system – actor interactions. A use case is a sequence of actions that provide a measurable value to an actor. Another way to look at it is a use case describes a way in which a real-world actor interacts with the system.

The overall goal of the system is to classify Kenyan sentiments obtained from Facebook and Twitter into three categories; positive, negative or neutral. It therefore has to provide interface for searching and fetching relevant Facebook and Twitter data. Diagrammatically, the system Use Case Model is as shown in figure below.

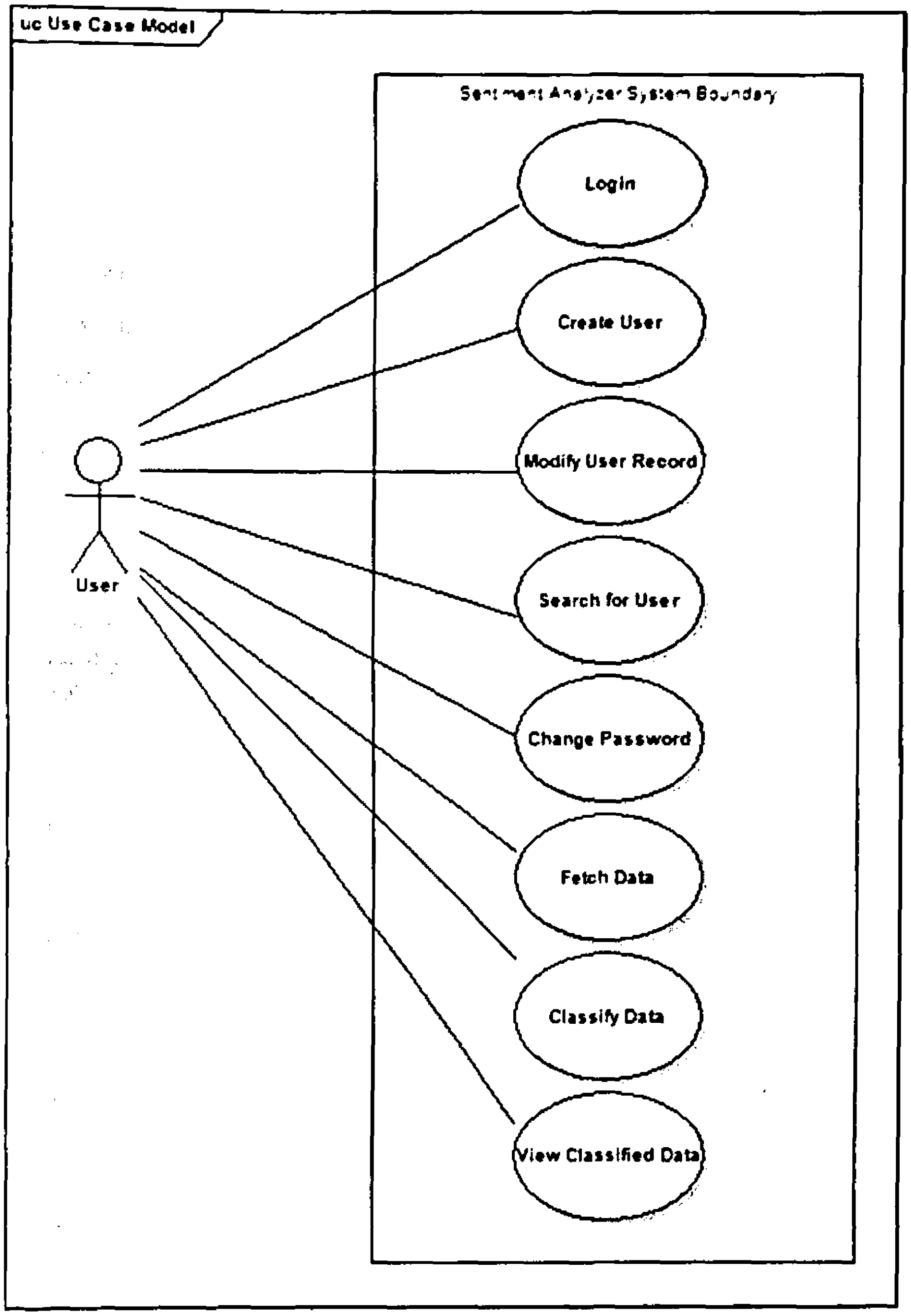


Figure 5. Use Case Model

4.1.2 Actors

Two actors are involved namely:-

i. The System

This refers to the Sentiment Analyzer System which constitutes various modules interconnected to provide various functionalities that fulfill the objectives of this research.

ii. User of the system

This is the human actor interested in benefiting from the utilities offered by the System functions.

4.1.3 Use Cases

4.1.3.1 Login/Authenticate User

Use Case ID :	UC-001				
Use Case Name :	Authenticate User				
Version:	**Version No.:**	**Created By**	**Date Created**	**Last Updated By:**	**Date Last Updated:**
	1.0	E. Wanyama	1/Apr/2012	E. Wanyama	1/Apr/2012
Actors :	User and Sentiment Analyzer System				
Description :	Authenticate User functionality allows the user associated with the use case to authenticate their user account to access system functionalities.				
Preconditions :	1. The account of the user has been created within the system. 2. The user has the right to access the system.				
Triggers :	The actor launches the application by launching a browser and entering the address of the application.				
Basic Course :	The following shall be the procedure to authenticate their account in the system:- 1. The actor launches the application by launching a browser and entering the address of the application. 2. System responds by availing login interface **UI1** 3. The user provides the user name and password for system authentication. 4. System validates credentials according to **UC-001-BR1** and acknowledges the user by displaying interface **UI2**. This indicates successful authentication to the system else **EP1, EP2, EP3 or EP4** apply. 5. **AP1** applies if it is the first login attempt and the password has not been changed.				

	6. End of system use case.
Alternative Paths (AP) :	1. System displays interface **UI6** for the actor to change password.
Post Conditions :	1. A log of the user authentication operation is stored. 2. User audit information, of the operation performed, stored by the system.
Exception Paths (EP) :	1. If actor has not entered user id a. System informs user "Invalid Username" b. System functionality remain unavailable to the user c. System logs the failed login attempt 2. If user id or password provided is invalid a. System notifies User with a message "Wrong Username or Password." b. System logs the failed login attempt c. System functionality remain unavailable to the user 3. If actor has not entered password System notifies user "Wrong Username or Password." 4. If actor's account is suspended a. System informs actor "User account is suspended, please contact administrator " b. System functionality remains unavailable to the User
Includes :	None
Priority :	High
Frequency of use :	Medium
Business Rules :	**UC-001-BR1: - Login validation** 1. Invalid users will not have access to the system. 2. The access to the functionalities of the system will be restricted to user authentication via user name and password.
Special Requirements:	None
Assumptions :	None
Process Owner :	Research Team

Notes and Issues :	None

4.1.3.2 Create User

Use Case ID :	UC-002				
Use Case Name :	Create User				
Version:	**Version No.:**	**Created By**	**Date Created**	**Last Updated By:**	**Date Last Updated:**
	1.0	E. Wanyama	1/Apr/2012	E. Wanyama	1/Apr/2012
Actors :	Super User, Sentiment Analyzer System				
Description :	Create User functionality allows the actor associated with the use case to add a new user to the system.				
Preconditions :	1. The actor has been authenticated and has access to the system. 2. The "Create User" function is available via the user interface.				
Triggers :	The User selects function: "Create User" from the System Administration Menu.				
Basic Course :	The following shall be the procedure to add a new user in the system:- 1. The User selects function: "Create User" from the System Administration Menu on interface **UI3**. 2. System displays **UI4** for capture of user details as specified in **UC-002-BR1**. 3. Actor saves the added user. 4. System validates entered details against rules specified in **UC-002-BR2**. Upon successful validation it assigns a password and user id to the added user's account and displays the password and user id to the actor. 5. End of system use case.				
Alternative Paths (AP) :	None				
Post Conditions :	1. Details of the new user stored in the database. 2. Log of the user creation operation updated. 3. User audit information, of the operation performed, stored by the system.				
Exception Paths (EP) :	1. If the actor has entered invalid data a) User prompted to enter correct data. 2. If actor has not captured all required details a) User prompted to capture all the required details 3. If the same user already exists within the system. a. User informed about the existence of the same user by a				

	message "the national id no. already exists in the system".
Includes :	UC-001
Priority :	Medium
Frequency of use :	Medium
Business Rules :	**UC-002-BR1: User details to be captured:-** i. National ID Number ii. Profile iii. Surname iv. Other Names v. Status (Active or Disabled) vi. Contact details Email address, physical address, postal address, postal code, Telephone and mobile number **UC-002-BR2: National Id Number** i. Length must be 8 characters ii. Must be numeric
Special Requirements:	None
Assumptions :	None
Process Owner :	Research Team
Notes and Issues :	The user interface shall provide input validation checks to ensure capture of valid data.

4.1.3.3 Modify User details

Use Case ID :	UC- 003				
Use Case Name :	Modify User details				
Version:	**Version No.:**	**Created By**	**Date Created**	**Last Updated By:**	**Date Last Updated:**
	1.0	E. Wanyama	1/Apr/2012	E. Wanyama	1/Apr/2012
Actors :	Super User, Sentiment Analyzer System				
Description :	Edit User functionality allows the actor associated with the use case to modify details of an existing user.				
Preconditions :	1. The actor has been appointed. 2. The actor's profile has the function to edit user. 3. The actor has been authenticated and has access to the system. 4. The list of existing users is available via the user interface. 5. The "Modify User details" function is available via the user interface.				

Triggers :	The actor selects function: "Update user details" from the System Administration Menu.
Basic Course :	The following shall be the procedure to edit a user in the system:- 1. The user selects "Modify user details" from the System Administration Menu on interface **UI3**. 2. System displays interface **UI7.** To facilitate location of user record 3. User enters search parameter for all or any of the attributes displayed and clicks on search button. 4. System displays list of users matching the search parameters entered. 5. Actor selects user record that needs to be updated from the list. 6. System displays user details on interface **UI5** in edit mode. 7. Actor makes changes as required and clicks on "Save" button to persist the changes. 8. System confirms successful save operation for the changes made on the user record. 9. End of system use case.
Alternative Paths (AP) :	
Post Conditions :	1. Changes to the user stored in the database.
Exception Paths (EP) :	1. If the actor has entered invalid data a) User prompted to enter correct data.
Includes :	UC-001, UC- 004
Priority :	Medium
Frequency of use :	Medium
Business Rules :	**UC- 003-BR1 allowed operations:-** 1. Actor is allowed to edit all attributes of the user except the National ID.
Special Requirements:	None
Assumptions :	None
Process Owner :	Research Team
Notes and Issues :	The user interface shall provide input validation checks to ensure capture of valid data.

4.1.3.4 Search User

Use Case ID :	UC- 004				
Use Case Name :	Search User				
Version:	**Version No.:**	**Created By**	**Date Created**	**Last Updated By:**	**Date Last Updated:**
	1.0	E. Wanyama	1/Apr/2012	E. Wanyama	1/Apr/2012
Actors :	User, Sentiment Analyzer System				
Description :	Search User functionality allows the actor associated with the use case to view a list of users matching specified search criteria.				
Preconditions :	1. The actor has been appointed. 2. The actor's profile has the function to search user. 3. The actor has been authenticated and has access to the system. 4. The "Search User" function is available via the user interface.				
Triggers :	Selection of "Modify User Details" or "Search User" functions from the System Administration menu.				
Basic Course :	The following shall be the procedure to search for an existing user in the system:- 1. Actor selects the "Search User" function from the System Administration menu. 2. System displays **UI7**. 3. Actor enters one or more user attributes to be used as search parameters and clicks on "Search" button to fetch results. 4. System displays a list of users whose records match the search parameters provided. If no search parameters are provided, system displays a list of all users. A link for drilling down to user details is provided beside each record in the "Choose" column. 5. Actor selects a user's record to view by clicking on the link in the "Choose" column. 6. System drills down to selected user details by displaying them in read only mode. 7. End of system use case.				
Alternative Paths (AP) :	None				
Post Conditions :	1. Log of the user viewing operation is stored. 2. User audit information, of the operation performed, stored by the system.				
Exception Paths (EP) :	1. If there are no users defined within the system. a) System informs the user about the unavailability of users in the system.				
Includes :	UC-001				
Priority :	Medium				

Frequency of use :	Medium
Business Rules :	
Special Requirements:	None
Assumptions :	None
Process Owner :	Research Team
Notes and Issues :	None

4.1.3.5 Change Password

Use Case ID :	SUC-005				
Use Case Name :	Change Password				
Version:	**Version No.:**	**Created By**	**Date Created**	**Last Updated By:**	**Date Last Updated:**
	1.0	E. Wanyama	1/Apr/2012	E. Wanyama	1/Apr/2012
Actors :	All authorized users				
Description :	Change Password functionality allows the actor associated with the use case to change password in the system.				
Preconditions :	1. The actor has an account created in the system. 2. The actor's profile has the function to change password. 3. The actor has been authenticated and has access to the system. 4. The function is available via the user interface.				
Triggers :	Selection of 'Change Password' function				
Basic Course :	The following shall be the procedure to reset the password of an existing user in the system:- 1. The actor navigates to the 'Change Password' function and selects it. 2. System provides interface **UI6** for actor to capture old and new passwords. 3. Actor provides the existing password and the new password. 4. The actor confirms the new password with the system. 5. System acknowledges successful change of password via "password changed successfully" message to the user. 6. End of system use case.				
Alternative Paths (AP) :	None				
Post Conditions :	1. User password information updated in the system. 2. A log of the password change operation is stored.				
Exception Paths (EP) :	1. If password is invalid a. System informs user to provide a valid password				

Includes :	UC-001
Priority :	High
Frequency of use :	medium
Business Rules :	
Special Requirements:	None
Assumptions :	None
Process Owner :	Research team
Notes and Issues :	

4.1.3.6 Fetch Data

Use Case ID :	UC-006				
Use Case Name :	Fetch Data				
Version:	**Version No.:**	**Created By**	**Date Created**	**Last Updated By:**	**Date Last Updated:**
	1.0	E. Wanyama	1/Apr/2012	E. Wanyama	1/Apr/2012
Actors :	User, Sentiment Analyzer System, Facebook and Twitter				
Description :	Fetch Data functionality allows the actor associated with the use case to search for and fetch desired data from Facebook and Twitter.				
Preconditions :	1. The actor has an account created in the system. 2. The actor's profile has the Fetch Data function. 3. The actor has been authenticated and has access to the system. 4. The function is available via the user interface. 5. The actor has obtained an access token from Facebook and fed it into the Sentiment Analyzer System through the Parameters interface.				
Triggers :	Actor selects "Fetch Data" function from the System administration menu.				
Basic Course :	The following shall be the procedure to fetch data from social media in the system:- 1. The actor navigates to the "Analyzer" menu and selects a "Fetch Data" function. 2. System displays interface **UI9** to facilitate searching of posts from Facebook and Twitter. 3. Actor enters search parameters as specified in **UC-006-BR1** and clicks on the Search button. 4. The system sends request for the data to Facebook and Twitter through the Graph API and Twitter API respectively. 5. Facebook and Twitter servers respond to the request by returning data in JSON format that meets the search Keyword(s) entered. 6. System decodes the JSON data returned into separate messages then displays them on the interface for the user to view. The message attributes fetched are as specified in **UC-006-BR2.**				

	7. Actor views the messages fetched and if satisfied clicks on the "Save" button to store them to the database. 8. System stores the messages and notifies the actor. 9. End of system use case.
Alternative Paths (AP) :	
Post Conditions :	1. Data of interest is stored in the database.
Exception Paths (EP) :	1. If no data is found then nothing is returned to the system. Actor adjust search parameters and repeats the search
Includes :	UC-001
Priority :	High
Frequency of use :	medium
Business Rules :	**UC-006-BR1: -Search parameters** 1. Keywords 2. Number of days (how far back in terms of days from the current date) **UC-006-BR2:- Message attributes** 1. Post Id 2. Message 3. Created time 4. Source (Facebook or Twitter)
Special Requirements:	None
Assumptions :	None
Process Owner :	Research Team
Notes and Issues :	

4.1.3.7 Classify Data

Use Case ID :	UC- 007				
Use Case Name :	Classify Data				
Version:	**Version No.:**	**Created By**	**Date Created**	**Last Updated By:**	**Date Last Updated:**
	1.0	E. Wanyama	6/Apr/2012	E. Wanyama	6/Apr/2012
Actors :	User, Sentiment Analyzer System and Python Classifier				
Description :	Classify Data functionality allows the actor associated with the use case to classify each of the messages in the data that was fetched and stored into three polarity categories negative, positive and neutral.				
Preconditions :	1. The actor has been authenticated and has access to the system. 2. The actor's profile has the function to Classify Data. 3. The function is available via the user interface.				
Triggers :	Actor selects the Classify function from the Analyzer menu.				
Basic Course :	The following shall be the procedure to classify data in the system:- 1. The actor navigates to the "Analyzer" menu then selects Classify Data function. 2. System displays interface **UI10** to facilitate classification operation. 3. User specifies messages to be classified by selecting search phrase related to them from a dropdown field then clicks on a 'View' button. 4. System displays unclassified messages for actor to view. 5. Actor then clicks on Classify Data button. 6. System calls the Python classifier module. 7. The classifier program performs its activities as specified in **UC-007-BR1**. 8. End of system use case.				
Alternative Paths (AP) :	None				
Post Conditions :	1. Polarity of each message is determined and stored for later display.				
Exception Paths (EP) :					
Includes :	UC-001				
Priority :	High				
Frequency of use :	High				
Business Rules :	**UC- 007-BR1** 1. Imports NLTK toolkit and other necessary tools 2. Reads in classifier training data from files where they are stored. 3. Extracts training features from the data read in. 4. Trains Naïve Bayes classifier 5. Connects to the database where Facebook and Twitter data are				

	stored. 6. Loops through each unclassified message/post stored as it classifies it and updates its polarity field.
Special Requirements:	None
Assumptions :	None
Process Owner :	Research Team
Notes and Issues :	

4.1.3.8 View Classification Results

Use Case ID :	UC-008				
Use Case Name :	View Results				
Version:	**Version No.:**	**Created By**	**Date Created**	**Last Updated By:**	**Date Last Updated:**
	1.0	E. Wanyama	6/Apr/2012	E. Wanyama	6/Apr/2012
Actors :	User, Sentiment Analyzer System				
Description :	View Results functionality allows the actor associated with the use case to view classification results in the system.				
Preconditions :	1. The actor's profile has the function to View Results. 2. The actor has been authenticated and has access to the system. 3. The function is available via the user interface.				
Triggers :	Actor selects "View results" function from the Analyzer menu.				
Basic Course :	The following shall be the procedure to View classification results in the system:- 1. The actor navigates to the "Analyzer" menu then select "View Results" function. 2. System displays interface **UI11** to facilitate viewing of results. 3. The actor selects a search phrase/keyword(s) then clicks on the "View data" button. 4. System displays a summary of classification results in a tabular format where the polarity is displayed in total percentages of each polarity category. System also displays each sentiment/message in a grid. The messages are displayed in a color corresponding to the individual polarity for ease of identification and viewing. 5. Actor clicks on "Generate pie chart" button to view a summary of the classification for the selected phrase. 6. System displays a pie chart showing classification summary. 7. End of system use case.				
Alternative Paths (AP) :	None				

Post Conditions :	1. User password information updated in the system. 2. A log of the password change operation is stored. 3. User audit information, of the operation performed, stored by the system.
Exception Paths (EP) :	None
Includes :	UC-001
Priority :	High
Frequency of use :	High
Business Rules :	
Special Requirements:	None
Assumptions :	None
Process Owner :	Research Team
Notes and Issues :	

4.2 Architectural Design

4.2.1 Architectural Overview

Figure 5 shows an overview of the system components and the interconnections between them that enable it to perform Sentiment analysis.

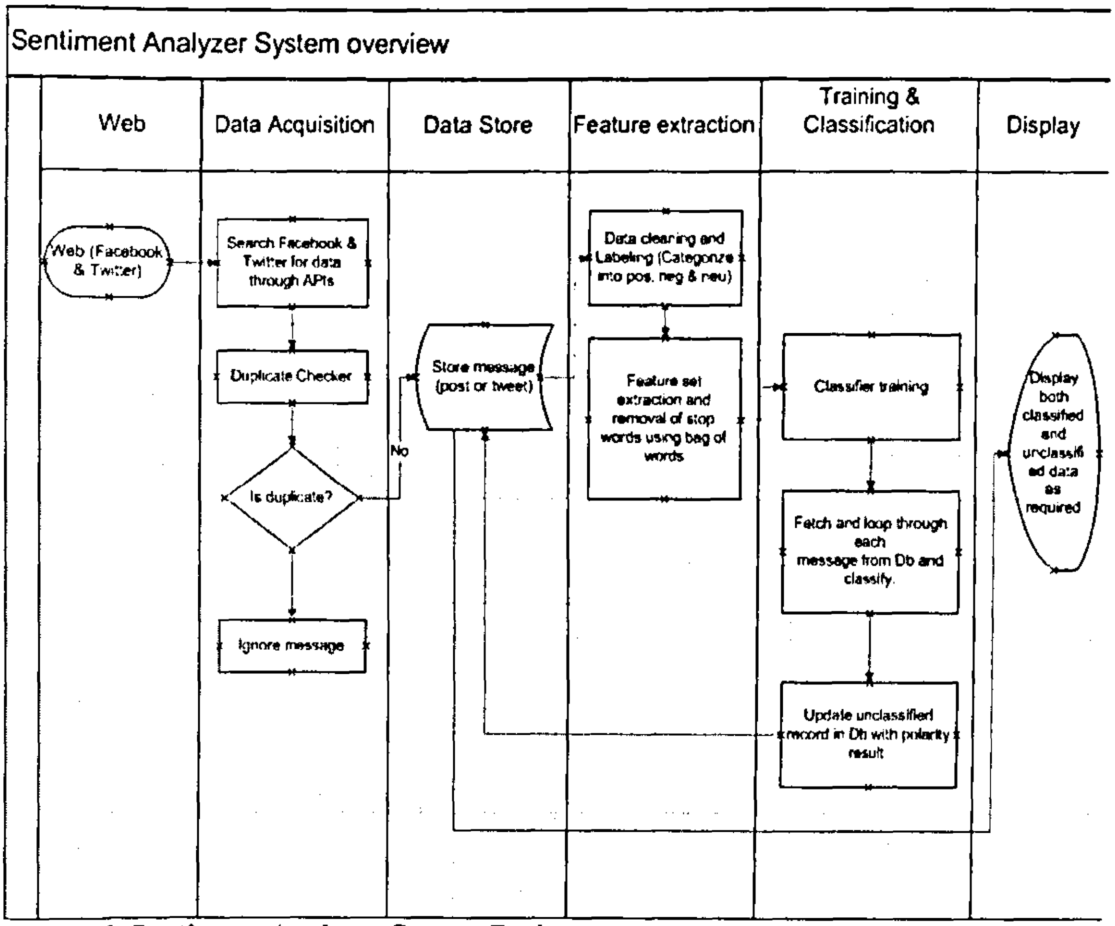

Sentiment Analyzer System overview
Web
Data Acquisition
Data Store
Feature extraction
Training & Classification
Display
Web (Facebook & Twitter)
Search Facebook & Twitter for data through APIs
Duplicate Checker
Is duplicate?
No
Ignore message
Store message (post or tweet)
Data clearing and Labeling (Categonze into pos. neg & neu)
Feature set extraction and removal of stop words using bag of words
Classifier training
Fetch and loop through each message from Db and classify.
Update unclassified record in Db with polarity result
Display both classified and unclassified data as required

Figure 6. Sentiment Analyzer System Design

Web: - The system interacts with the Web in order to fetch data from Facebook and Twitter

Facebook and Twitter APIs: - These APIs provide an interface for interaction between Sentiment Analyzer System and the two social networks.

Duplicate Checker: - The purpose of this function in the Data acquisition module is to ensure there is no duplication of messages fetched from Facebook and Twitter. It uses the message Ids to enforce this check.

Data Store: - The purpose of this module is to store data fetched from Twitter and Facebook into a MySQL database.

Feature Extraction Module: - This module does extraction of feature sets which are rendered to the classifier in 'feature - label' pairs where the feature is the word and the label is the polarity. It also removes stop words which have been found to be insignificant in the classification task. It is called into action throughout the classification process.

Training and Classification Module: - This module is charged with training a Naïve Bayes Classifier based on the features returned by the feature extractor. The trained classifier is then used to determine the polarity of sentiments in the database.

Display Module: - This module provides visualizations for the data classification results in form of percentages and Charts.

4.3 System Implementation

The Sentiment Analyzer System was implemented as a three tier application using Java Enterprise Edition tools as given below:-

4.3.1 Front End

Java Server Faces (JSF) was used in development of forms that constitute the user interfaces/forms.

AJAX – AJAX was used in input validations on user interfaces

Richfaces – we also used rich face components such as text

Cascading Style Sheets (CSS):- this was used in managing styles for the interfaces. An example of CSS usage is in the coloring scheme used to display the three polarities. We used red color for negative sentiments, blue for neutral while green denotes positive sentiments.

4.3.2 Application Logic/Middle tier

The business logic was implemented using Java programming Language. This involved definition of entity properties into Plain Old Java Objects (Pojos) whereas the logic was done in Java Bean classes.

4.3.3 Backend

MySQL database was used to implement backend objects which include tables, relationships, constraints and sequences. Hibernate was used to provide the requisite mapping between the database and the application logic.

4.3.4 Classifier Module

This module was implemented using Python programming language and the Natural Language Toolkit (NLTK) tools. We customized the Naïve Bayes Classifier that is available in the NLTK toolkit. Part of NLTK data was also used in providing a set of stop words. As earlier explained the feature set function checks and removes any appearance of stop words from the sentiments being classified.

4.3.5 Data Model

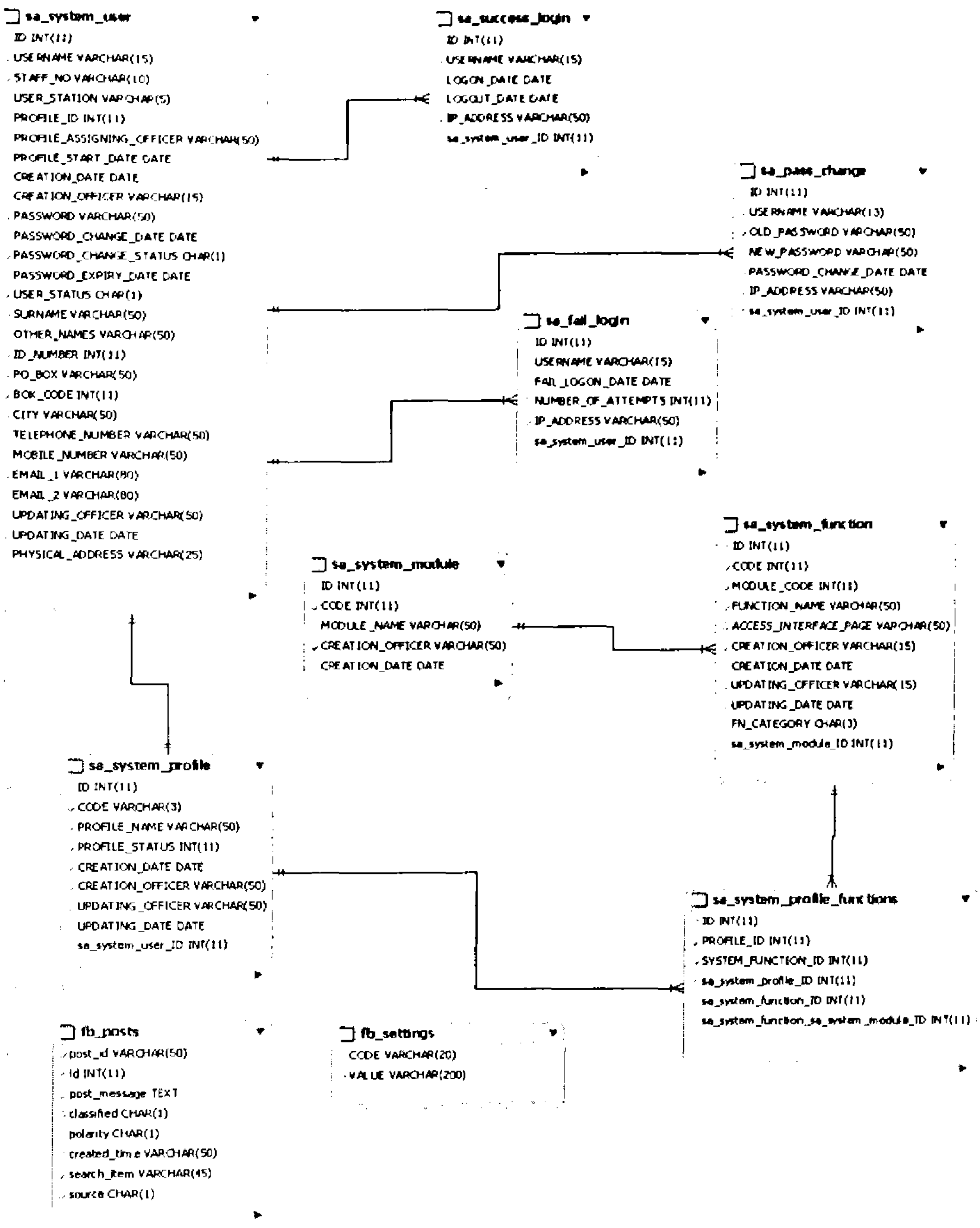


Figure 7. Sentiment Analyzer System Data Model

4.4Summary of Methodology

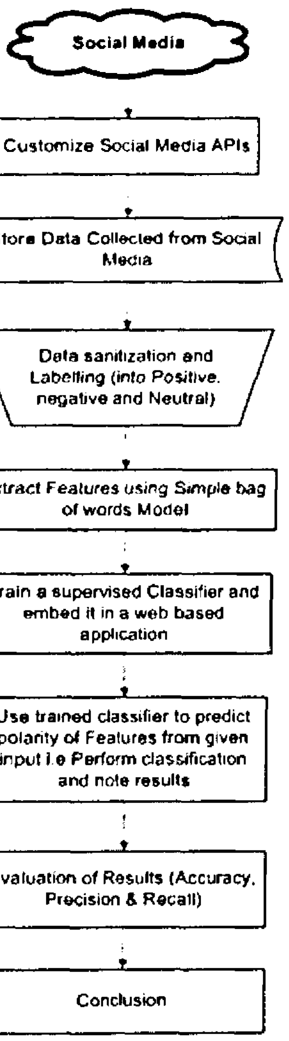

Social Media
Customize Social Media APIs
Store Data Collected from Social Media
Data sanitization and Labelling (into Positive, negative and Neutral)
Extract Features using Simple bag of words Model
Train a supervised Classifier and embed it in a web based application
Use trained classifier to predict polarity of Features from given input i.e Perform classification and note results
Evaluation of Results (Accuracy, Precision & Recall)
Conclusion

Figure 8. Summary of Methodology Used

CHAPTER 5

5.0 Results and Findings

In this section, we present the results of our experimentation in this research work. We performed two sets of tests, one set for checking the functionalities of the entire application which was done using the Test Cases (i) to (vii) given under appendix 3 while the other was done with the Naïve Bayes Classifier. In experimenting with the Naïve Bayes Classifier, we relied on the NLTK metrics module which provides functions for calculating accuracy, precision and recall for the Classifier.

5.1 Test runs and Presentation of Results

We performed a number of tests whose results are as tabulated below:-

5.1.1 Test runs with Naïve Bayes Classifier

5.1.1.1 Experiments with unbalanced corpus

i. **Effect of stop words**

Experiments with unbalanced corpus (negative set – 1036 words, neutral set – 367 words, positive set – 610 words) without removing low information gain features.								
No.	**Experiment**	**Classifier Accuracy %**	**Pos. Precision %**	**Pos. Recall %**	**Neg. Precision %**	**Neg. Recall %**	**Neu. Precision %**	**Neu. Recall %**
1.	Classification with stop words	77.2	63.7	77.9	87.0	92.2	71.7	35.9
2.	Classification without stop words	76.4	62.6	77.9	87.8	91.1	65.3	34.8

Table 2. Effect of stop words on unbalanced corpus

ii. Effect of low information gain features without removing stop words

Experiments with unbalanced corpus (negative set – 1036 words, neutral set – 367 words, positive set – 610 words) when low information gain features are removed.								
No.	Experiment	Classifier Accuracy %	Pos. Precision %	Pos. Recall %	Neg. Precision %	Neg. Recall %	Neu. Precision %	Neu. Recall %
1.	With best 500 words	77.6	66.4	69.3	83.0	95.5	79.6	42.4
2.	With best 1000 words	77.8	67.1	71.4	83.3	95.1	79.2	41.3
3.	With best 1500 words	78.2	65.6	73.6	85.8	95.5	76.6	39.1
4.	With best 1950 words	78.9	68.2	73.6	85.1	95.5	76.6	40.9
5.	With best 2000 words	78.2	65.6	73.6	85.8	95.5	76.6	39.1

Table 3. Effect of removing low information gain features with stop words retained

5.1.1.2 Experiments with balanced corpus

i. Effect of stop words

Experiments with balanced corpus (negative set – 350 words, neutral set – 350 words, positive set – 350 words) without removing low information gain features.								
No.	Experiment	Classifier Accuracy %	Pos. Precision %	Pos. Recall %	Neg. Precision %	Neg. Recall %	Neu. Precision %	Neu. Recall %
1.	Classification with stop words	54.5	44.8	87.5	73.3	37.5	72.3	38.6
2.	Classification without stop words	53.8	45.3	83.0	73.2	34.1	62.9	44.3

Table 4. Effect of stop words on balanced corpus

ii. **Effect of low information gain features without removing stop words**

Experiments with balanced corpus (negative set – 350 words, neutral set – 350 words, positive set – 350 words) when low information gain features are removed.								
No.	**Experiment**	**Classifier Accuracy %**	**Pos. Precision %**	**Pos. Recall %**	**Neg. Precision %**	**Neg. Recall %**	**Neu. Precision %**	**Neu. Recall %**
1.	With best 500 words	58.7	46.0	84.1	86.1	35.2	74.6	56.8
2.	With best 700 words	58.7	46.0	84.1	86.1	35.2	74.6	56.8
3.	With best 800 words	58.7	46.0	84.1	86.1	35.2	74.6	56.8
4.	With best 900 words	59.5	46.9	85.2	82.5	37.5	76.6	55.7
5.	With best 1000 words	61.0	48.1	85.2	84.1	42.0	76.6	55.7
6.	With best 1050 words	61.4	48.1	85.2	84.8	44.3	77.4	54.5

Table 5. Effect of removing low information gain features on balanced corpus

5.1.2 Sentiment Analyzer System results

The sentiment analyzer system was tested using test cases (see appendix 3). All the system features were observed to work as stipulated in the design.

5.2 Evaluation of Results

The results presented in part (i) of section 4.1.1.1 above indicate that removal of stop words lowers the performance of the classifier on all the metrics except negative precision which went up by 0.8%.

We further see from the results given in parts (ii) of sections 4.1.1.1 and 4.1.1.2 above that removal of low information features improves the performance of the classifier on all the three metrics (i.e. accuracy, precision and recall) for all the corpus sizes used in experimentation.

We also tested the effect of using a balanced corpus for training where we provided an equal number of words for positive, negative and neutral words. A general drop in performance measures was observed which improved with the increase in the corpus size.

We observe that on average the best classifier model is achieved when we used a modest corpus of 1950 most informative words which gave highest classifier accuracy of 78.9%. Other metrics also appear to perform at an acceptable level when all results are considered.

5.3 Discussion of Results

The evaluation results in this study suggest that feature reduction steps such as removal of stop words should be taken very carefully since their removal could have a negative impact on classifier model as observed in this study. Some stop words are highly discriminative features.

We deduce that out of the hundreds of features our classification model has some of them had low information. These are features that are common across negative, positive and neutral classes, and therefore contribute little information to the classification process. Individually these features are harmless, but in aggregate we observed from the results given in parts (ii) of sections 4.1.1.1 and 4.1.1.2 above that low information features can decrease performance. Thus eliminating low information features gave our model clarity by removing noisy data. This explains why the use of only the higher information features, led to an increase in performance while also decreasing the size of the model.

We also interpret the following from an accuracy of 78.9%, Positive precision of 68.2%, Positive recall of 73.6%, Negative precision of 85.1%, Negative recall of 95.5%, Neutral Precision of 76.6% and Neutral recall of 40.9 %:-

1. A sentiment given a positive classification is only 68.2% likely to be correct. This precision leads to 31.8% false positives for the positive label.
2. 73.6% positive recall means that there is a likelihood of 26.4% of getting false negatives and false neutrals in the positive class.
3. A sentiment identified as negative is 85.1% likely to be correct. This means a 14.9% likelihood of having false positives and neutrals in the negative class.
4. A negative recall of 95.5% means that many sentiments that are negative are correctly classified. This implies very few false negatives for the negative category.
5. A sentiment given a neutral classification is only 76.6% likely to be correct. This precision leads to 23.4% false neutrals for the neutral label.
6. A neutral recall of 40.9% means that there is a high likelihood of 59.1% of getting false negatives and false positives in the neutral class.

One possible explanation for the above results is that people use normally positive words in negative reviews, but the words are preceded by a negating word such as "not" (or some other negative word), such as "not great". And since the classifier uses the bag of words model, which assumes every word is independent, it cannot learn that "not great" is a negative.

CHAPTER 6

Conclusion and Future Work

Conclusion

In this research work we were able to build an application that has the ability to fetch and store data searched on various products and services from two most popular social media sites (Facebook and Twitter). A number of classification models were considered as specified in the literature review out of which we chose to use the Naïve Bayes classifier model because of its simplicity in adapting it to the data collected. We developed a Naïve Bayes classifier that integrates an information gain heuristic using the Natural Language Tool Kit and trained it on a preprocessed dataset from Social Media.

The results obtained from experiments with the classifier (see Chapter 5 above) show that the classifier is capable of performing classification with an accuracy of 78.9% for sentiments obtained from Social Media. This is near human accuracy, as apparently people agree on sentiment only around 80% of the time. Most of the sentiments in this data are expressed partly in English, Swahili, Slang and Sheng thus formal language is scarcely used. We therefore conclude that the model of classification selected is ideal for the kind of data collected from social media on Kenyan products and services.

Finally, we integrate the techniques and methods developed into a web based application for use in providing Sentiment Analysis with respect to products and services in Kenya. The application provides user friendly features and its architecture can also be used in viewing results of other classification approaches.

Limitations Faced

A number of limitations were faced as listed:-

1. We were not able to fetch more information associated with sentiments fetched from social media due to restrictions that prevent running of queries that have joins.

2. While fetching data, the requests are limited to a time span of 30 seconds beyond which a fetch operation is timed out and disconnected. This limits the amount of data that can be fetched on a search keyword.
3. The use of positive words preceded by negations such as 'not' in negative sentiments leads to erroneous classifications since the classifier uses the bag of words model, which assumes every word is independent. It cannot therefore learn that "not great" is a negative.
4. Based on the data obtained from Kenyan opinions we observed that the language used is mostly slang. Kenyan slang is constantly changing coupled with the idea of lack of inadequate literature on it since it is informal. This made it challenging during preparation of training data and also during classification.
5. We were challenged in finding a way of filtering data from Facebook to Kenyan content since Facebook restricts running of queries with joins. Consequently, some of the keywords searched for would fetch sentiments done in irrelevant and unknown languages
6. We were also limited to 7 days and 14 days at most in terms of how far back the search results would go for Twitter and Facebook respectively.

Future Work

The following improvements can be done on the Sentiment Analyzer System:-

1. Functionalities for accommodating other classifiers other than the Naïve Bayes classifier can be developed into the application. These classifiers include Decision trees and Support Vector Machines. Results from the various classifiers can be compared in a report interface for the best classification technique to be selected.
2. Training on multiple words can also be explored to resolve the limitation 3 faced in section 6.2 above.
3. The application can also be enhanced to be accessible on handheld devices such as mobile phones.
4. Integration with multi and cross-lingual language dictionaries to cater for the dynamic nature of language used on social media.

References

Abbasi, A., Chen, H., and Salem A. (2007). *Sentiment Analysis in Multiple Languages: Feature Selection for Opinion Classification in Web Forums.* Available at: http://ai.arizona.edu/intranet/papers/ahmedabbasi_sentimenttois.pdf. Last accessed 9th Nov 2011.

Bird S., Klein E., and Loper E., (2009). *Natural Language Processing with Python.* 1st ed. CA: O'Reilly Media.

Bonchi, F., Castillo, C., Gionis, A., and Jaimes, A. (2011). *Social network analysis and mining for business applications.* Available: http://doi.acm.org/10.1145/1961189.1961194. Last accessed 10th Oct 2011.

Boisen, S. (2008). *Natural Language Processing in Python using NLTK.* Available at: http://semanticbible.com/other/talks/2008/nltk/main.html. Last accessed 9th Aug 2011.

Corbo, R. (2012). *The History of Social Media Evolution on the Internet.* Available: http://www.technize.com/the-history-of-social-media-evolution-on-the-internet. Last accessed 18th Apr 2011.

Cui, H., Mittal, V., and Datar, M. (2006). Comparative experiments on sentiment classification for online product reviews, In Proceedings of *AAAI.*

Facebook Team. (2012). *Facebook Platform Policies.* Available at: http://developers.facebook.com/policy. Last accessed 11th Jan 2012.

Gitau, E., and Miriti, E. (2011). *An approach for Using Twitter to perform Sentiment Analysis in Kenya,* University of Nairobi, Kenya.

Hu, M., and Liu, B. 2004. Mining and summarizing customer reviews. In *Proceedings of KDD '04, the ACM SIGKDD international conference on Knowledge discovery and data mining,* 168–177. Seattle, US: ACM Press.

Liu, B. (2010). *Sentiment Analysis: A multi-faceted Problem* . Available at: http://www.cs.uic.edu/~liub/FBS/IEEE-Intell-Sentiment-Analysis.pdf. Last accessed 8th Nov 2011.

Mayeku, D. (2011). *Top 10 Social Networking Sites in Kenya December 2011.* Available at: http://www.openbook.co.ke/2011/top-10-social-networking-sites-in-kenya-december-2011.html. Last accessed 19th Apr 2012.

Mejova, Y. (2009). *Sentiment Analysis – an overview.* Available at: http://www.divms.uiowa.edu/~ymejova/publications/CompsYelenaMejova.pdf. Last accessed 1st Nov 2011.

Neil, G. (2011). *Sentiment Analysis and Social Media Marketing.* Available at: http://socialtimes.com/sentiment-analysis-social-media-marketing1_b58996. Last accessed 8th Nov 2011.

Pak, A. and Paroubek, P., 2009. *Twitter as a Corpus for Sentiment Analysis and Opinion Mining*, Universit´e de Paris-Sud, Cedex, France

Pang, B., Lee, L. and Vaithyanathan, S, (2002). Thumbs up? Sentiment Classification Using Machine Learning Techniques. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing(EMNLP).* pp.79–86

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

Socialbakers. (2012). *Kenya Facebook Statistics.* Available at: http://www.socialbakers.com/facebook-statistics/kenya#chart-intervals. Last accessed 19th Apr 2012.

Wel, L. and Royakkers, L. (2004). *Ethical Issues in Web data mining.* Available: http://alexandria.tue.nl/repository/freearticles/612259.pdf. Last accessed 7th Nov 2011.

Yessenov, K. and Misailovic, S. (2009). *Sentiment Analysis of Movie Review Comments.* Available: http://people.csail.mit.edu/kuat/courses/6.863/report.pdf . Last accessed 13th Jul 2011.

Appendix

Appendix 1 – Sample Source Code

Classifier Source Code

```
###### TASK           - SENTIMENT ANALYZER FOR FACEBOOK  POSTS
###### FILE NAME      - fb_classifier.py
###### AUTHOR         - EDWIN WANYAMA NGERO
###### ADMISSION NO.- P58/72924/2009
###Train classifier using data in file .
###Then use trained classifier to classify posts fetched from facebook.

#***************Import objects required.
import nltk
import MySQLdb
import collections, itertools
import nltk.classify.util, nltk.metrics
import re
import nltk.classify.util
import nltk.data
#import unicodedata
from nltk.classify import NaiveBayesClassifier
#from nltk.corpus import movie_reviews
from nltk.corpus import stopwords
from nltk.corpus.reader import CategorizedPlaintextCorpusReader
from nltk.probability import FreqDist, ConditionalFreqDist
from nltk.metrics import BigramAssocMeasures

#****************Train the classifier using data stored in files then
return message when classification is complete.

#********Function for extracting words into a dictionary as it also
removes english stopwords

stopset = set(stopwords.words('english'))
#def stopword_filtered_word_feats(words):
#    return dict([(word, True) for word in words])

def word_feats(words):
    return dict([(word, True) for word in words])
#################

#======== read in cleaned facebook and twitter data
cleanreader    =    CategorizedPlaintextCorpusReader('C:/Documents    and
Settings/edwin/nltk_data/corpora/clean', r'(pos|neg|neu)/.*\.txt',
cat_pattern=r'(pos|neg|neu)/(\w+)\.txt')

cleannegids = cleanreader.fileids('neg')[:1000]
cleanposids = cleanreader.fileids('pos')[:600]
cleanneuids = cleanreader.fileids('neu')[:350]

word_fd = FreqDist()
label_word_fd = ConditionalFreqDist()
```

```
for word in cleanreader.words(categories=['pos']):
    word_fd.inc(word.lower())
    label_word_fd['pos'].inc(word.lower())

for word in cleanreader.words(categories=['neg']):
    word_fd.inc(word.lower())
    label_word_fd['neg'].inc(word.lower())

for word in cleanreader.words(categories=['neu']):
    word_fd.inc(word.lower())
    label_word_fd['neu'].inc(word.lower())

pos_word_count = label_word_fd['pos'].N()
neg_word_count = label_word_fd['neg'].N()
neu_word_count = label_word_fd['neu'].N()
total_word_count = pos_word_count + neg_word_count + neu_word_count

word_scores = {}

for word, freq in word_fd.iteritems():
    pos_score = BigramAssocMeasures.chi_sq(label_word_fd['pos'][word],
        (freq, pos_word_count), total_word_count)
    neg_score = BigramAssocMeasures.chi_sq(label_word_fd['neg'][word],
        (freq, neg_word_count), total_word_count)
    neu_score = BigramAssocMeasures.chi_sq(label_word_fd['neu'][word],
        (freq, neu_word_count), total_word_count)
    word_scores[word] = pos_score + neg_score + neu_score

best    =    sorted(word_scores.iteritems(),    key=lambda    (w,s):    s,
reverse=True)[:1950]
bestwords = set([w for w, s in best])

def best_word_feats(words):
    return dict([(word, True) for word in words if word in bestwords])


cleannegfeats  =  [(best_word_feats(cleanreader.words(fileids=[f])),  'neg')
for f in cleannegids]
cleanposfeats  =  [(best_word_feats(cleanreader.words(fileids=[f])),  'pos')
for f in cleanposids]
cleanneufeats  =  [(best_word_feats(cleanreader.words(fileids=[f])),  'neu')
for f in cleanneuids]

cleannegcutoff = len(cleannegfeats)*3/4
cleanposcutoff = len(cleanposfeats)*3/4
cleanneucutoff = len(cleanneufeats)*3/4

trainfeats          =          cleannegfeats[:cleannegcutoff]          +
cleanposfeats[:cleanposcutoff]+cleanneufeats[:cleanneucutoff]
testfeats          =           cleannegfeats[cleannegcutoff:]          +
cleanposfeats[cleanposcutoff:]+cleanneufeats[cleanneucutoff:]

print 'train on %d instances, test on %d instances' % (len(trainfeats),
len(testfeats))

classifier = NaiveBayesClassifier.train(trainfeats)
refsets = collections.defaultdict(set)
```

```
testsets = collections.defaultdict(set)

def stopword_filtered_word_feats(words):
    return dict([(word, True) for word in words if word not in stopset])
#######################
#print 'evaluating best word features when using only the best 1950 words'
for i, (feats, label) in enumerate(testfeats):
    refsets[label].add(i)
    observed = classifier.classify(feats)
    testsets[observed].add(i)

print     'pos     precision:',     nltk.metrics.precision(refsets['pos'],
testsets['pos'])
print 'pos recall:', nltk.metrics.recall(refsets['pos'], testsets['pos'])
print     'pos     F-measure:',     nltk.metrics.f_measure(refsets['pos'],
testsets['pos'])
print     'neg     precision:',     nltk.metrics.precision(refsets['neg'],
testsets['neg'])
print 'neg recall:', nltk.metrics.recall(refsets['neg'], testsets['neg'])
print     'neg     F-measure:',     nltk.metrics.f_measure(refsets['neg'],
testsets['neg'])
print     'neu     precision:',     nltk.metrics.precision(refsets['neu'],
testsets['neu'])
print 'neu recall:', nltk.metrics.recall(refsets['neu'], testsets['neu'])
print     'neu     F-measure:',     nltk.metrics.f_measure(refsets['neu'],
testsets['neu'])
print 'accuracy:', nltk.classify.util.accuracy(classifier, testfeats)
classifier.show_most_informative_features()

print '*********End of evaluating best word features'

#**********Establish database connection parameters:-

from nltk.tokenize import word_tokenize

try:
#*********create a connection object.
 connection  =  MySQLdb.connect(host="localhost",  user="root",  passwd="",
db="test" )

#*********create a cursor.
 cursor = connection.cursor()

#*********Fetch data i.e. post_id and post_message.
 cursor.execute(  "SELECT  post_id,  post_message  FROM  fb_posts  where
classified ='N'")

#*********get the resultset as a tuple
 result = cursor.fetchall()

#*********Iterate through each post message as you:-
#**************preprocess(remove unwanted characters and stop words) then
classify it.
 for record in result:
  #print record[1]
  postid = record[0]
  message = record[1]
  message = message.translate(None, '$%*#?~_!.+\',')
  #print message
  tokenized_msg = word_tokenize(message.strip().lower())

   #**********Assign the results of the classification, i.e. the label to
a variable.
  polarityrslt = classifier.classify(word_feats(tokenized_msg))
  if polarityrslt == 'pos':
   polrslt = 'P'
    #print polarityrslt
  elif polarityrslt == 'neg':
   polrslt = 'N'
  elif polarityrslt == 'neu':
   polrslt = 'W'

  sqlstmnt = ("""UPDATE fb_posts SET polarity = %s WHERE post_id = %s""",
(polrslt, postid))
  #print sqlstmnt

  #***************Using the post_id as identifier, Update the database
record with the classification result(label)
  cursor.execute("Update fb_posts set polarity = %s, classified = %s where
post_id = %s", (polrslt, 'Y', postid))
  connection.commit()

  #**********Trap any erros
except MySQLdb.OperationalError, message:
 errorMessage = "Error %d:\n%s" % (message[ 0 ], message[ 1 ] )
 connection.rollback()
finally:
 cursor.close()
 connection.close()
```

AnalyzerBean.java code

```
package ics.wanyama.analyzer;

import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import ics.common.UseMessageBundle;
import ics.common.converters.DateConverter;
import ics.common.data.DataAccessService;
import ics.common.data.HibernateUtil;
import ics.common.data.PersistentDataEntity;
import ics.common.ui.ParentBean;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;
import javax.faces.model.SelectItem;
import javax.imageio.ImageIO;
import javax.servlet.http.HttpServletRequest;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JRExporterParameter;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
import net.sf.jasperreports.engine.export.JRXhtmlExporter;
import org.hibernate.Hibernate;
import org.hibernate.Query;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.general.DefaultPieDataset;
import org.jfree.data.general.PieDataset;
import com.restfb.types.Post;
import test.PieChart3DDemo1;
public class AnalyzerBean extends ParentBean {
      FacesContext context = FacesContext.getCurrentInstance();
      HttpServletRequest request = (HttpServletRequest) context
                  .getExternalContext().getRequest();
      String remoteHost = request.getRemoteHost();
      private FbPost currentPost = new FbPost();
      private       List<PersistentDataEntity>      posts      =      new
      ArrayList<PersistentDataEntity>();
      private int positive = 0;
      private int negative = 0;
      private int neutral = 0;
      private String accessToken;
      private int days;
      private String keyWord;
      private boolean showGraph = false;
      private      ArrayList<SelectItem>      searchCriteria      =      new
      ArrayList<SelectItem>();
      private int numberOfPosts = 0;
      public AnalyzerBean() {
      }
      public void setPosts(List<PersistentDataEntity> posts) {
            this.posts = posts;
      }
      public List<PersistentDataEntity> getPosts() {
            return posts;
      }
      public void viewClasifiedData(ActionEvent e) {
            keyWord                         =                         (String)
            e.getComponent().getAttributes().get("key_word");
            posts.clear();
            HibernateUtil.getCurrentSession().clear();
            posts = DataAccessService
            .entityListFromHQL("from FbPost  f  where  f.searchItem  =  '"  +
            keyWord + "' and f.classified='Y'");
            System.out.println("KEy word ->" + keyWord);
            System.out.println("Number of posts ->" + posts.size());
```

```
        setShowGraph(false);
        setPositive(0);
        setNegative(0);
        setNeutral(0);
        for (PersistentDataEntity post : posts) {
              if (((FbPost) post).getPolarity().equalsIgnoreCase("P"))
                    positive++;
              else                    if                    (((FbPost)
              post).getPolarity().equalsIgnoreCase("N"))
                    negative++;
              else                    if                    (((FbPost)
              post).getPolarity().equalsIgnoreCase("W"))
                    neutral++;
        }
  }
  public void viewUnClassifiedData(ActionEvent e) {
        keyWord                        =                     (String)
        e.getComponent().getAttributes().get("key_word");
        posts.clear();
        posts = DataAccessService
        .entityListFromHQL("from FbPost f where f.searchItem = '"
        + keyWord + "' and f.classified='N'");
        System.out.println("KEy word ->" + keyWord);
        System.out.println("Number of posts ->" + posts.size());
        System.out.println(".........Test          posts         ->"+
        DataAccessService.entityListFromHQL("from    FbPost    f   where
        f.searchItem = 'WATER' and f.classified='N'").size());
        setShowGraph(false);
        setPositive(0);
        setNegative(0);
        setNeutral(0);
        for (PersistentDataEntity post : posts) {
              if (((FbPost) post).getPolarity().equalsIgnoreCase("P"))
                    positive++;
              else                    if                    (((FbPost)
              post).getPolarity().equalsIgnoreCase("N"))
                    negative++;
              else                    if                    (((FbPost)
              post).getPolarity().equalsIgnoreCase("W"))
                    neutral++;
        }
  }
  public void showGraph(ActionEvent e) {
        generateLineGraph();
        setShowGraph(true);
        try {
  FacesContext.getCurrentInstance().getExternalContext().redirect(
                          "/icloud/analyzer/viewData.faces");
        } catch (IOException e1) {
              // TODO Auto-generated catch block
              e1.printStackTrace();
        }
  }
  public void fetchData(ActionEvent e) {
        posts.clear();
        this.currentPost.setSubmitMessage("");
        String accessToken = ((Settings) DataAccessService
                .findFirst("from   Settings   s   where   s.code  =
        'access_token'"))
                .getValue();
        keyWord                         =                        ((String)
        e.getComponent().getAttributes().get("key_word"))
                .replace(" ", "+");
        days = (Integer) e.getComponent().getAttributes().get("days");

        try {
            FbUtility   fbUtility   =   new   FbUtility(accessToken,
        keyWord, days);
            System.out.println("............created fbUtility");
            fbUtility.runEverything();
            System.out
                    .println("............
        fbUtility.runEverything() successful");
            posts.addAll(fbUtility.getFetchedPosts());
        } catch (Exception ex) {
            System.out.println("............fetch data failed");
            ex.printStackTrace();
        }
    }
    public void classifyData(ActionEvent e) {
        keyWord                         =                        (String)
        e.getComponent().getAttributes().get("key_word");
        runPythonScript();
        posts.clear();
        setPositive(0);
        setNegative(0);
        setNeutral(0);
        HibernateUtil.getCurrentSession().clear();
        System.out.println("..........Test posts ->"
         +  DataAccessService.entityListFromHQL("from  FbPost  f  where
        f.searchItem = 'WATER' and f.classified='N'").size());
        HibernateUtil.getCurrentSession().flush();
        posts.clear();
        posts = DataAccessService
        .entityListFromHQL("from FbPost f where f.searchItem = '"
                        + keyWord + "'");
        System.out.println("KEy word ->" + keyWord);
        System.out.println("Number of posts ->" + posts.size());
        setShowGraph(false);
        setPositive(0);
        setNegative(0);
        setNeutral(0);
        for (PersistentDataEntity post : posts) {
            if (((FbPost) post).getPolarity().equalsIgnoreCase("P"))
                positive++;
            else                if                 (((FbPost)
            post).getPolarity().equalsIgnoreCase("N"))
                negative++;
            else                if                 (((FbPost)
            post).getPolarity().equalsIgnoreCase("W"))
                neutral++;
        }
    }
    public void runPythonScript() {
```

```
		try {
			String line;
			String	argsToPythonInterpreter	=	"cmd	/c
			C:\\fb_classifier.py";
			Process wanyama = Runtime.getRuntime()
					.exec(argsToPythonInterpreter);
			BufferedReader	bri	=	new	BufferedReader(new
			InputStreamReader(
					wanyama.getInputStream()));
			BufferedReader	bre	=	new	BufferedReader(new
			InputStreamReader(
					wanyama.getErrorStream()));
			while ((line = bri.readLine()) != null) {
				System.out.println(line);
			}
			bri.close();
			while ((line = bre.readLine()) != null) {
				System.out.println(line);
			}
			bre.close();
			wanyama.waitFor();
			System.out.println("Done.");
		} catch (Exception err) {
			err.printStackTrace();
		}
	}
	public void setCurrentPost(FbPost currentPost) {
		this.currentPost = currentPost;
	}
	public FbPost getCurrentPost() {
		return currentPost;
	}
	public void saveFetchedPosts() {
		try {
			HibernateUtil.beginTransaction();
			for (PersistentDataEntity fbp : posts) {
				if (((FbPost) fbp).getMessage() != null
						&&				((FbPost)
fbp).getMessage().length() <= 200)
					HibernateUtil.save((FbPost) fbp);
			}
			HibernateUtil.commitTransaction();
			this.currentPost.setSubmitMessage("Posts		saved
Successfully");
		} catch (Exception e) {
			e.printStackTrace();
			this.currentPost
					.setSubmitMessage("An	error	occurred	in
trying to save the posts");
		}
	}
	public void generateChartMethod(OutputStream out, Object data) {
		System.out.println("...........generating chart");
		System.out.println("...........Positive		->"		+
this.getPositive());
		System.out.println("...........Negative		->"		+
this.getNegative());
```

```
            System.out.println("...........Neutral          ->"         +
this.getNeutral());
            PieChart3DDemo1 chart1 = new PieChart3DDemo1("Summary");

            PieDataset  dataset  =  createSampleDataset(this.getPositive(),
this
                        .getNegative(), this.getNeutral());
            JFreeChart   chart   =   ChartFactory.createPieChart("Summary",
dataset, true, true, true); // urls
            BufferedImage  buffImg   =   chart.createBufferedImage(500,   //
width
                        375, // height
                        BufferedImage.TYPE_INT_RGB, // image type
                        null);
            try {
                  ImageIO.write(buffImg, "jpeg", out);
            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
      }
      private PieDataset  createSampleDataset(int  positive,  int  negative,
      int neutral) {
            final DefaultPieDataset result = new DefaultPieDataset();
            result.setValue("Negative", negative);
            result.setValue("Neutral", neutral);
            result.setValue("Positive", positive);
            return result;
      }
      public int getPositive() {
            return positive;
      }
      public void setPositive(int positive) {
            this.positive = positive;
      }
      public int getNegative() {
            return negative;
      }
      public void setNegative(int negative) {
            this.negative = negative;
      }
      public int getNeutral() {
            return neutral;
      }
      public void setNeutral(int neutral) {
            this.neutral = neutral;
      }
      public void setAccessToken(String accessToken) {
            this.accessToken = accessToken;
      }
      public String getAccessToken() {
            return accessToken;
      }
      public void setDays(int days) {
            this.days = days;
      }
      public int getDays() {
```

```
		return days;
	}
	public String getKeyWord() {
		return keyWord;
	}
	public void setKeyWord(String keyWord) {
		this.keyWord = keyWord;
	}
	public int getNumberOfPosts() {
		return this.getPosts().size();
	}
	public void setSearchCriteria(ArrayList<SelectItem> searchCriteria)
{
		this.searchCriteria = searchCriteria;
	}
	public ArrayList<SelectItem> getSearchCriteria() {
		return searchCriteria;
	}
	public void setShowGraph(boolean showGraph) {
		this.showGraph = showGraph;
	}
	public boolean isShowGraph() {
		return showGraph;
	}
	private List<ChartData> trendlData = new ArrayList<ChartData>();
	public static JasperPrint jasperPrint;
	private String rootPath = UseMessageBundle.getMessageResourceString(
			FacesContext.getCurrentInstance().getApplication()
					.getMessageBundle(),
"application_root_physical_path",
			null,
FacesContext.getCurrentInstance().getViewRoot().getLocale());
	public void generateLineGraph() {
		trendlData = generateData(2012, this.getKeyWord());
		JRBeanCollectionDataSource	beanCollectionDataSource	=	new
JRBeanCollectionDataSource(
					trendlData);
		try {
			jasperPrint = JasperFillManager.fillReport(rootPath
					.concat("analyzer/line	graph.jasper"),	new
HashMap(),
					beanCollectionDataSource);
		} catch (JRException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		JRXhtmlExporter xhtmlExporter = new JRXhtmlExporter();
	xhtmlExporter.setParameter(JRExporterParameter.JASPER_PRINT,
				jasperPrint);
	xhtmlExporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME,
				rootPath.concat("analyzer/lineGraph.xhtml"));
	xhtmlExporter.setParameter(JRExporterParameter.IGNORE_PAGE_MARGINS,r
ue);
		try {
			xhtmlExporter.exportReport();
		} catch (JRException e) {
			e.printStackTrace();
		}
	}
	public List<ChartData> generateData(int year, String searchItem) {
		List<ChartData> data = new ArrayList<ChartData>();
		ChartData c;
		int totalNumber = 1000;
		int data1 = 0;
		Query datesQuery = DataAccessService.session().createQuery(
		"select  distinct  MONTH(f.createdTime)  from  FbPost  f  where
f.searchItem = '" + searchItem + "' order by MONTH(f.createdTime) asc");

		ArrayList<Integer> datesList = new ArrayList<Integer>();
		for (Iterator it = datesQuery.iterate(); it.hasNext();) {
			datesList.add((Integer) it.next());
		}
		System.out.println("............Number   of   days   ->"   +
		datesList.size());
		for (Integer d : datesList) {
			c = new ChartData();
			c.setDay(d.toString());
			c.setPositive(DataAccessService.entityListFromHQL(
			"from FbPost f where MONTH(f.createdTime) ='" + d
			+ "' and f.polarity ='P' and f.searchItem = '"
			+ searchItem + "'").size());
			c.setNegative(DataAccessService.entityListFromHQL(
			"from FbPost f where MONTH(f.createdTime) ='" + d
			+ "' and f.polarity ='N' and f.searchItem = '"
			+ searchItem + "'").size());
			c.setNeutral(DataAccessService.entityListFromHQL(
			"from FbPost f where MONTH(f.createdTime) ='" + d
			+ "' and f.polarity ='W' and f.searchItem = '"
			+ searchItem + "'").size());
			data.add(c);
			System.out.println("............day ->" + c.getDay()
					+ "...... positive ->" + c.getPositive()
					+ "...... negative ->" + c.getNegative()
					+ "...... neutral ->" + c.getNeutral());
		}
		return data;
	}
}
```

ChartData.java

```
package ics.wanyama.analyzer;
import ics.common.data.PersistentDataEntity;
public class ChartData extends PersistentDataEntity {

	private String day;
	private Integer positive;
	private Integer negative;
	private Integer neutral;
	public ChartData() {
	}
	public String getDay() {
		return day;
	}
	public void setDay(String day) {
		this.day = day;
	}
	public Integer getPositive() {
		return positive;
	}
	public void setPositive(Integer positive) {
		this.positive = positive;
	}
	public Integer getNegative() {
		return negative;
	}
	public void setNegative(Integer negative) {
		this.negative = negative;
	}
	public Integer getNeutral() {
		return neutral;
	}
	public void setNeutral(Integer neutral) {
		this.neutral = neutral;
	}
}
```

FbPost.java

```
package ics.wanyama.analyzer;
import ics.common.data.PersistentDataEntity;
import ics.common.general.Address;
import java.util.*;
import javax.faces.context.FacesContext;
public class FbPost extends PersistentDataEntity{

	private int id;
	private String postId;
	private String message;
	private String classified;
	private String polarity;
	private Date createdTime;
	private String searchItem;
	private String source;

	public FbPost(){
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getPostId() {
		return postId;
	}
	public void setPostId(String postId) {
		this.postId = postId;
	}
	public String getMessage() {
		return message;
```

```
    }
    public void setMessage(String message) {
          this.message = message;
    }
    public String getClassified() {
          return classified;
    }
    public void setClassified(String classified) {
          this.classified = classified;
    }
    public String getPolarity() {
          return polarity;
    }
    public void setPolarity(String polarity) {
          this.polarity = polarity;
    }
    public Date getCreatedTime() {
          return createdTime;
    }
    public void setCreatedTime(Date createdTime) {
          this.createdTime = createdTime;
    }
    public void setSearchItem(String searchItem) {
          this.searchItem = searchItem;
    }
    public String getSearchItem() {
          return searchItem;
    }
    public void setSource(String source) {
          this.source = source;
    }
    public String getSource() {
          return source;
    }
}
```

FbUtility.java

```
package ics.wanyama.analyzer;
import static java.lang.System.currentTimeMillis;
import static java.lang.System.out;
import ics.common.data.PersistentDataEntity;
import ics.wanyama.analyzer.FbPost;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.List;
import javax.el.ELContext;
import javax.faces.context.FacesContext;
import org.jfree.io.IOUtils;
import winterwell.json.JSONArray;
import winterwell.json.JSONObject;
import com.restfb.Connection;
import com.restfb.DefaultFacebookClient;
import com.restfb.FacebookClient;
import com.restfb.Parameter;
import com.restfb.types.Post;
import com.restfb.types.User;

public class FbUtility {

	private final FacebookClient facebookClient;
	private List<FbPost> fetchedPosts = new ArrayList<FbPost>();
	private String keyWord = "";
	private int days = 1;
	public static void main(String[] args) {new FbUtility(
	"AAACEdEose0cBAJoJUALxZBlWGTRlDOCmpGNGvEhEwUXzx3uRHfpgSnUmw7qrd6EfQv
MVdPwKa7ZB9lgyTevUlFGwA38ablNe6d5jmCyunntpgDHQI0",    "water",
1).runEverything();
	}
	FbUtility(String accessToken, String key_word, int entered_days) {
		this.keyWord = key_word;
		this.days = entered_days;
		facebookClient = new DefaultFacebookClient(accessToken);
	}
	void runEverything() {
		search();
		selection();
		parameters();
		rawJsonResponse();
	}
	void search() {
		fetchedPosts.clear();
		Date periodStart = addDaysFromDate(new Date(), 0 - days);
		Connection<Post>                publicSearch                =
facebookClient.fetchConnection("search",  Post.class,  Parameter.with("q",
this.keyWord),  Parameter.with("type",  "post"),  Parameter.with("since",
	periodStart),Parameter.with("limit",100    ),Parameter.with("locale",
"en_US"));
		for (Post p : publicSearch.getData()) {
			FbPost fbp = new FbPost();
			fbp.setPostId(p.getId());
			fbp.setMessage(p.getMessage());
			fbp.setCreatedTime(p.getCreatedTime());
			fbp.setClassified("N");
			fbp.setPolarity(null);
			fbp.setSource("F");
			fbp.setSearchItem(keyWord.replace("+"," "));
			if        (fbp.getMessage()       !=       null       &&
fbp.getMessage().length() <= 200)
				fetchedPosts.add(fbp);
		}
		out.println("Number     of     fetched     posts:     "     +
fetchedPosts.size());
		//Twitter posts
		System.out.println("...................Started on twitter");

		String username = "google";   // update
		// get tweets as JSON
```

```
            URL url = null;
            try {
                  url = new URL("http://search.twitter.com/search.json?q="
+ keyWord );
            } catch (MalformedURLException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
            ByteArrayOutputStream       urlOutputStream       =       new
ByteArrayOutputStream();
            try {
                  IOUtils.getInstance().copyStreams(url.openStream(),
urlOutputStream);
            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
            String urlContents = urlOutputStream.toString();
      System.out.println(urlContents.substring(urlContents.indexOf("result
s")+9));
            System.out.println(urlContents);
            // parse JSON
            JSONArray               jsonArray               =               new
JSONArray(urlContents.substring(urlContents.indexOf("results")+9));
            // use
            for (int i = 0; i < jsonArray.length(); i++) {
                  JSONObject jsonObject = jsonArray.getJSONObject(i);
                  System.out.println(jsonObject.getString("id"));
                  System.out.println(jsonObject.getString("text"));
                System.out.println(jsonObject.getString("created_at"));
                  FbPost fbp = new FbPost();
                  fbp.setPostId(jsonObject.getString("id"));
                  fbp.setMessage(jsonObject.getString("text"));
                  fbp.setCreatedTime(new
~~Date~~(jsonObject.getString("created_at")));
                  fbp.setClassified("N");
                  fbp.setPolarity(null);
                  fbp.setSource("T");
                  fbp.setSearchItem(keyWord);
                  System.out.println("...........Twitter post complete" +
jsonObject.getString("id"));
                  if (fbp.getMessage() != null)
                        fetchedPosts.add(fbp);
            }
      }
      void selection() {
            out.println("* Selecting specific fields *");
            User   user   =   facebookClient.fetchObject("me",   User.class,
Parameter.with("fields", "id,name"));
            out.println("User name: " + user.getName());
      }
      void parameters() {
            out.println("* Parameter support *");
            Date periodStart = addDaysFromDate(new Date(), 0 - days);
            Connection<Post>                filteredFeed                =
 facebookClient.fetchConnection("me/feed",                  Post.class,
 Parameter.with("limit", 300), Parameter.with("since", periodStart));
```

```
            out.println(".........SInce: " + periodStart);
            out.println(".........Filtered     feed     count:     "     +
filteredFeed.getData().size());
        }
        void rawJsonResponse() {
            out.println("* Raw JSON *");
            out.println("User object JSON: "
                        + facebookClient.fetchObject("me", String.class));
        }
        public void setFetchedPosts(List<FbPost> fetchedPosts) {
            this.fetchedPosts = fetchedPosts;
        }
        public List<FbPost> getFetchedPosts() {
            return fetchedPosts;
        }
        public static Date addDaysFromDate(Date enteredDate, int noOfDays) {
            GregorianCalendar           gc           =            new
GregorianCalendar(enteredDate.~~getYear~~(),
                        enteredDate.~~getMonth~~(), enteredDate.~~getDate~~());
            gc.add(Calendar.DATE, noOfDays);
            gc.add(Calendar.YEAR, 1900);
            return gc.getTime();
        }
}
```

Settings.java

```
package ics.wanyama.analyzer;
import ics.common.data.PersistentDataEntity;
import ics.common.general.Address;
import java.util.*;
import javax.faces.context.FacesContext;
public class Settings extends PersistentDataEntity{

        private String code;
        private String value;
        public Settings(){
        }
        public String getCode() {
            return code;
        }
        public void setCode(String code) {
            this.code = code;
        }
        public String getValue() {
            return value;
        }
        public void setValue(String value) {
            this.value = value;
        }

}
```

analyzer.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
        <class          lazy="false"         name="ics.wanyama.analyzer.FbPost"
table="fb_posts">
                <id column="ID" name="id" type="integer">
                        <generator class="increment" />
                </id>
                <property generated="never" name="postId" type="string"
                        column="POST_ID" />
                <property generated="never" name="message" type="string"
                        column="POST_MESSAGE" />
                <property generated="never" name="classified" type="string"
                        column="CLASSIFIED" />
                <property generated="never" name="polarity" type="string"
                        column="POLARITY" />
                <property generated="never" name="createdTime" type="date"
                        column="CREATED_TIME" />
                <property generated="never" name="searchItem" type="string"
                        column="SEARCH_ITEM" />
                <property generated="never" name="source" type="string"
                        column="SOURCE" />
        </class>
        <class name="ics.wanyama.analyzer.Settings" table="fb_settings">
                <id name="code" column="CODE" type="string" />
                <property name="value" column="VALUE" type="string" />
        </class>

</hibernate-mapping>
```

Appendix 2 – Installation Instructions

a) **JDK Installation instructions.**

1. Search, download and Install JDK 1.6.0 or a higher and compatible version from the internet.
2. Go to programs>Right click My Computer and select properties. Then click on advanced tab menu.
3. Under user variables section, click on new button to setup a new user variable. Provide the name as "JAVA_HOME" and provide the variable value as "C:\Program Files\Java\jdk1.6.0" i.e. the path to the bin folder where JDK was installed (Note that the "bin" name should not be put at the end). Click Ok to add the variable.
4. Go to System variables section and select the Path variable and click the "Edit" button.
5. Check under variable value to see if the path "C:\Program Files\Java\jdk1.6.0\bin" exists. Add the path on the list incase it does not exist. Remember to separate it from the previous paths by use of a semicolon.
6. Click OK to save the new variable.
7. Close the environment variables and system properties dialog.
8. Test Java installation by using the commad below on windows command prompt:

```
java -version
```

The system will respond with a java version message if java is correctly installed.

b) **Tomcat Application Server Installation Instructions.**

1. Download windows installer for Tomcat and install it as a windows service. Use the checkbox on the component page to set the service as "auto" startup, so that Tomcat is automatically started when Windows starts.

c) Python Tools Installation Instructions.

1. Navigate to folder named Python tools on installation CD. You see a number of tools to be installed.
2. Install the tools in the following order by double clicking on each:
   a. Python 2.6.6
   b. numpy-1.5.1-win32-superpack-python2.6
   c. matplotlib-1.0.1.win32-py2.6
   d. PyYAML-3.09.win32-py2.6
   e. nltk-2.0b9.win32
3. Go to programs and launch Python 2.6.6. IDLE(Python GUI)
4. Go to file > Open and browse to CD. Select customdatapath_creator.py program file and run it to create a custom nltk data path.
5. Copy folder named "corpora" with all its contents from installation CD and paste it under the "nltk_data" folder (Check for it under C:/Documents and Settings/<user>/).
6. Copy file named fb_classifier.py from installation CD and paste it under drive C i.e. C://fb_classifier.py.
7. Open Control Panel and go to Tools > folder options. Select File types tab then click on "new" button to set a new file association type. In the "Create new extension" dialog that pops up, enter "py" and specify the associated file type as "Python".

d) MySQL Server Database Installation Instructions.

1. Download and install MySQL database software
   NB: Leave user "root" without password during installation.
2. Download and install HeidiSQL software to help in deploying, managing databse objects.
3. Navigate to folder named Python tools on installation CD. You see a number of tools to be installed.
4. Double click "MySQL-python-1.2.3c1.win32-py2.6" to install the Python - MySQL connector.
5. Follow instructions as prompted by the Wizard to instal to completion.

6. Launch HeidiSQL and login as user "root" then ensure you are in "test" database and open the Query tab. Open and Copy "database scripts" file from installation CD then paste them into the Heidi Query window and execute/run to deploy database objects.

e) **Sentiment Analyzer System Installation Instructions.**

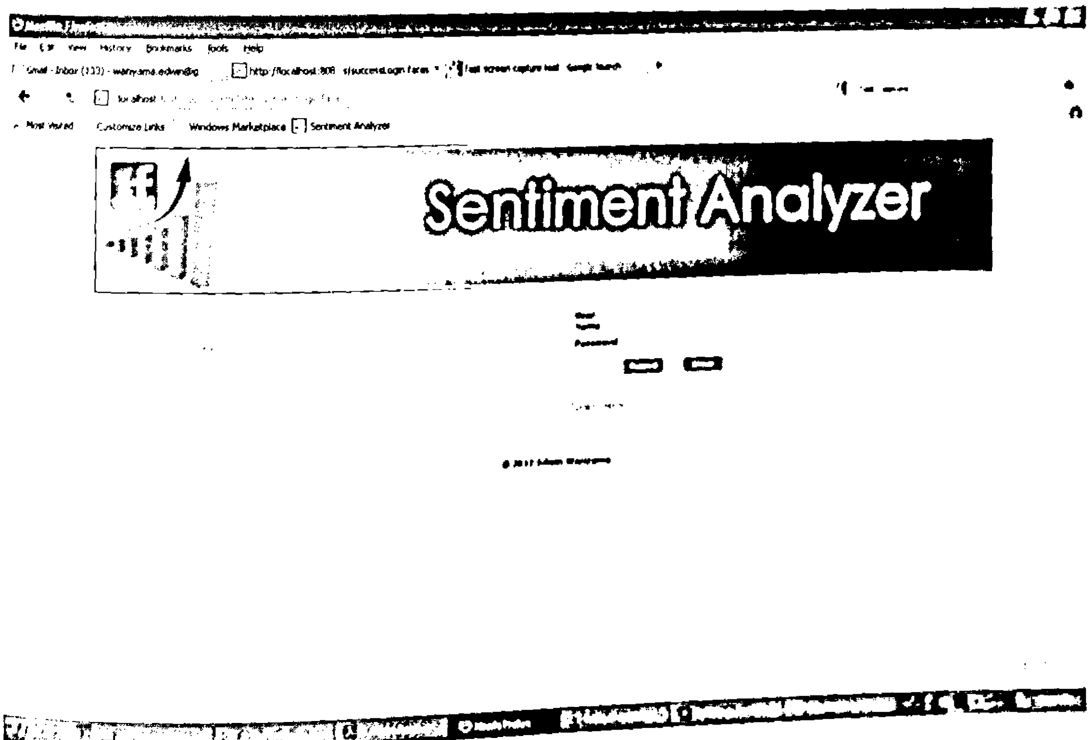
1. Copy folder named "wanyama" from CD and paste it under webapps folder in apache-tomcat installation directory.
2. Restart Apache-tomcat application server. The restart ensures application is deployed.

Appendix 3 - User manual for sentiment Analyzer System

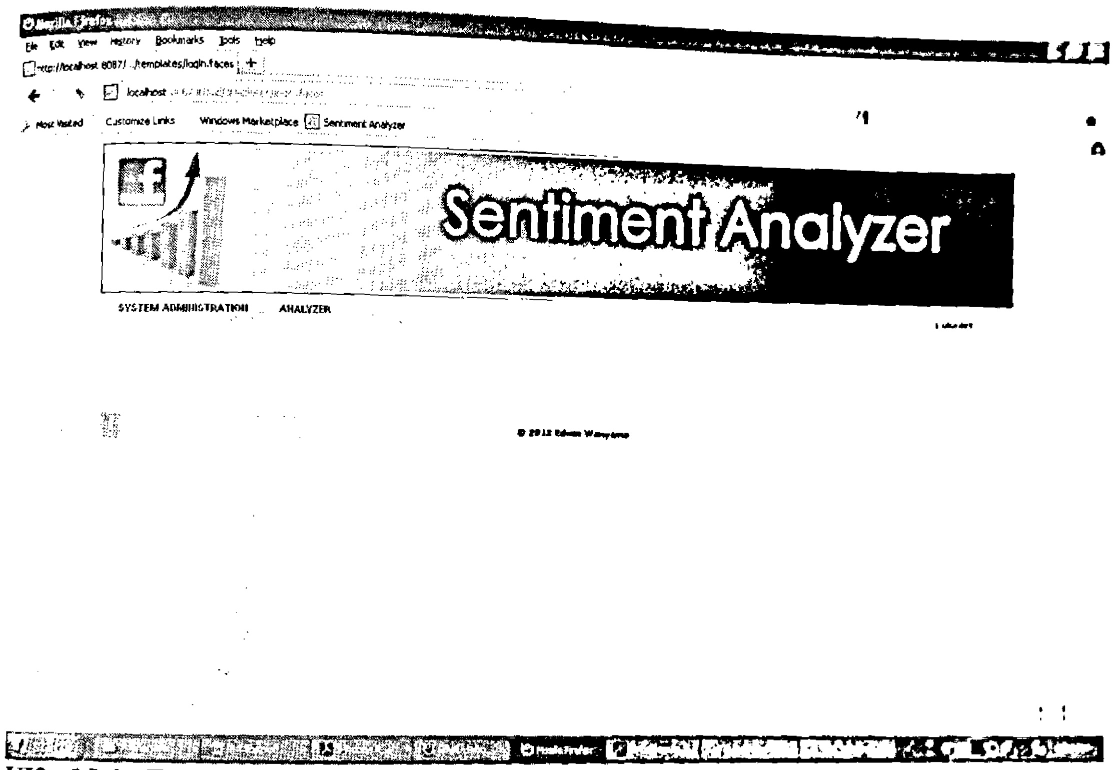
Login

The following are steps to be followed during login to the application :

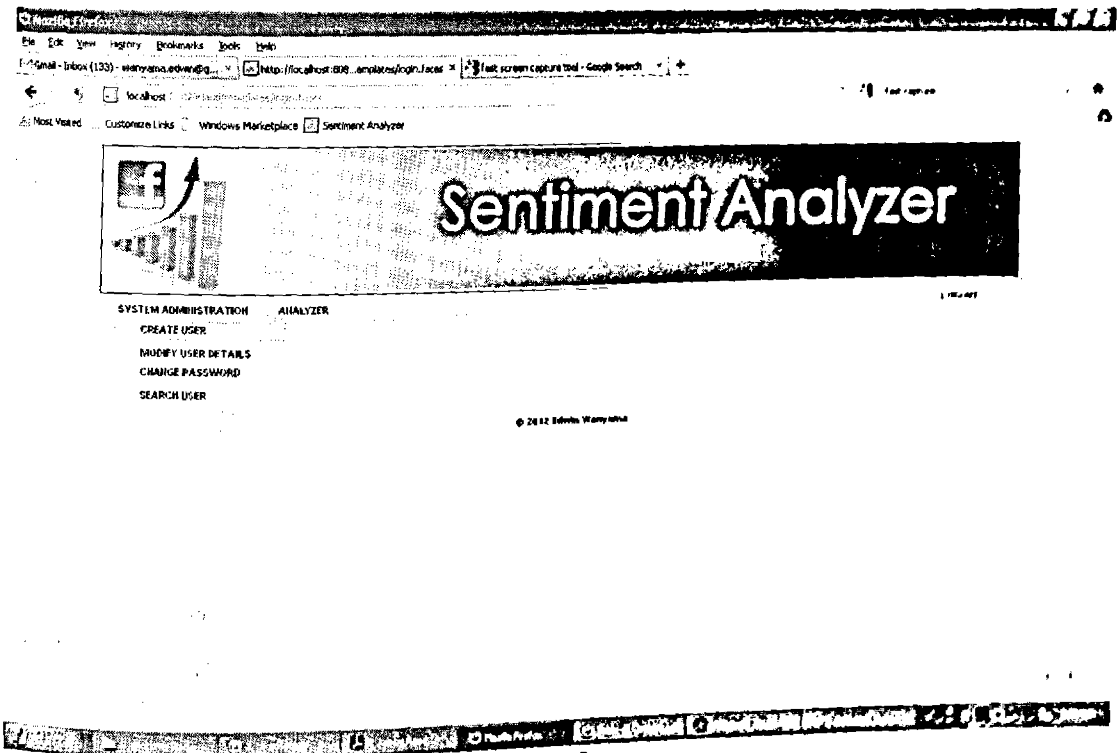
1. Launch your browser and enter URL given below and press enter http://localhost:8087/icloud/templates/login.faces (if installed on the same computer) or http://XXXX:8087/icloud/templates/login.faces (where XXXX is the IP of the application server)
2. System will present login form shown as UI1. Enter username and password then click on the 'Submit' button to login.
3. To clear the username and password fields of entered data click on 'Clear' button
4. If login credentials are correct system presents interface UI2 which is the Main Form See interface below.
5. The main menus and their respective sub menus are visible on interface UI2, UI3 and UI8



UI1 - Login Form



UI2 - Main Form

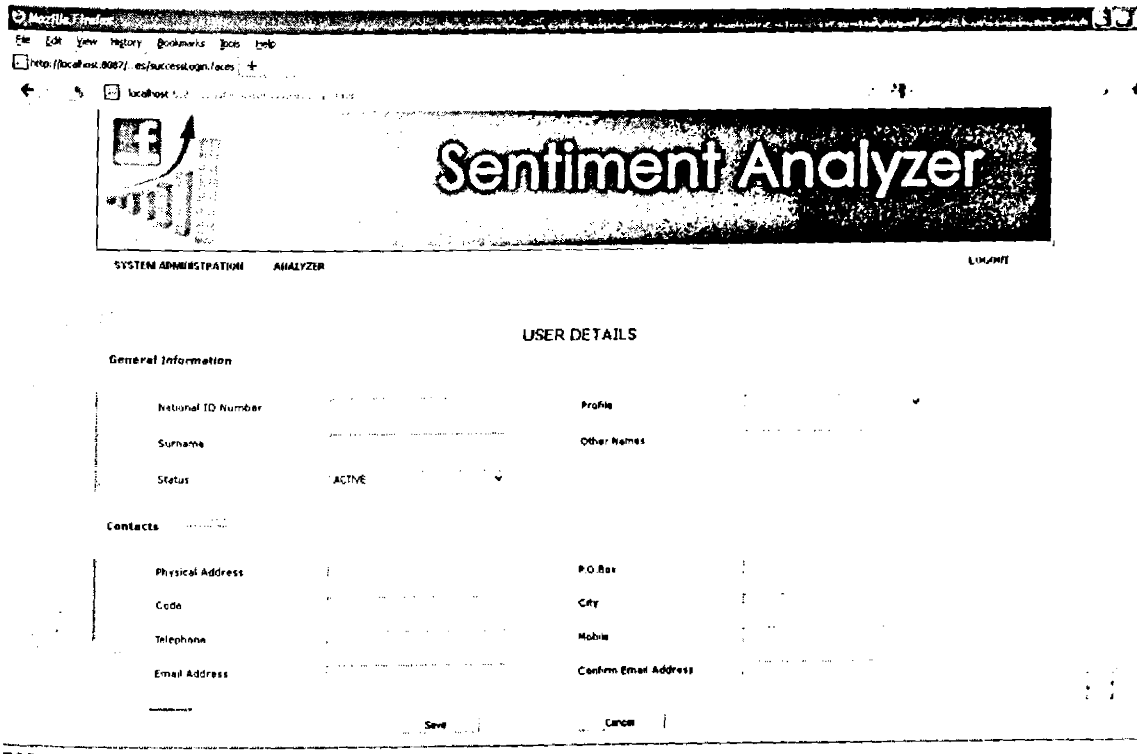


UI3 - System Administration Menu and Menu Items

Create User

The following are the steps to be followed in creating a new user in the system:-

1. Login to the system in case you are not logged.
2. Navigate to the 'SYSTEM ADMINISTRATION' menu and select 'CREATE USER' submenu as shown above.
3. System will respond by displaying interface UI4 for use in capturing new user details.
4. Capture user details as labeled on the interface then click on the 'Save' button to save the new user's details.
5. System saves new user record and creates user id and password that are displayed to the user.
6. Send the login credentials to the new user.
7. To exit from this interface click on the 'Cancel' button.



USER DETAILS

General Information

National ID Number		Profile	
Surname		Other Names	
Status	ACTIVE		

Contacts

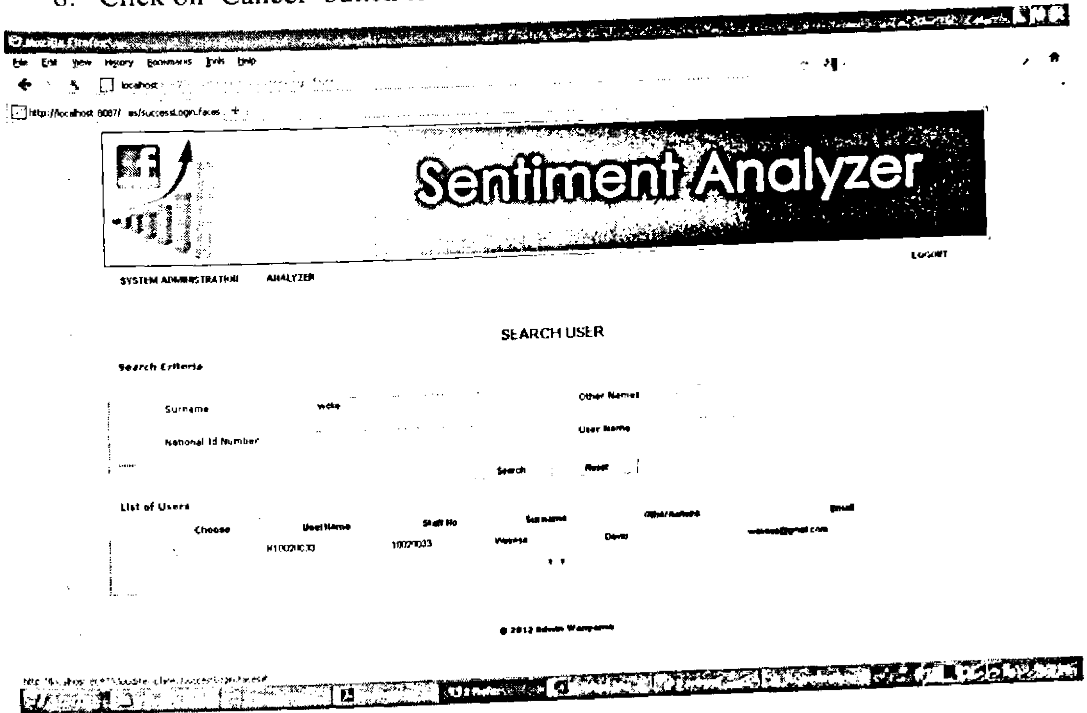
Physical Address		P.O.Box	
Code		City	
Telephone		Mobile	
Email Address		Confirm Email Address	

UI4 - Create User Form

Modify User Details

To modify a user's details in the system do the following:-

1. While logged into the system navigate to the 'SYSTEM ADMINISTRATION' menu and select the 'MODIFY USER DETAILS' submenu.
2. System displays interface **UI7 – Search User Form.**
3. Enter search criteria then click 'Search' button to locate the user of interest.
4. System will respond with a list of users meeting search criteria provided otherwise no results will be returned.
5. Select the user record of interest by clicking on the 'Choose' icon on the extreme left column of the record.
6. System will respond with a display of the user details in edit mode. The display is same as **UI4 – create User Form.**
7. Make changes to the user record and click on 'Save' button to persist the changes.
8. Click on 'Cancel' button to exit.



Surname	weke
Other Names	
National Id Number	
User Name	

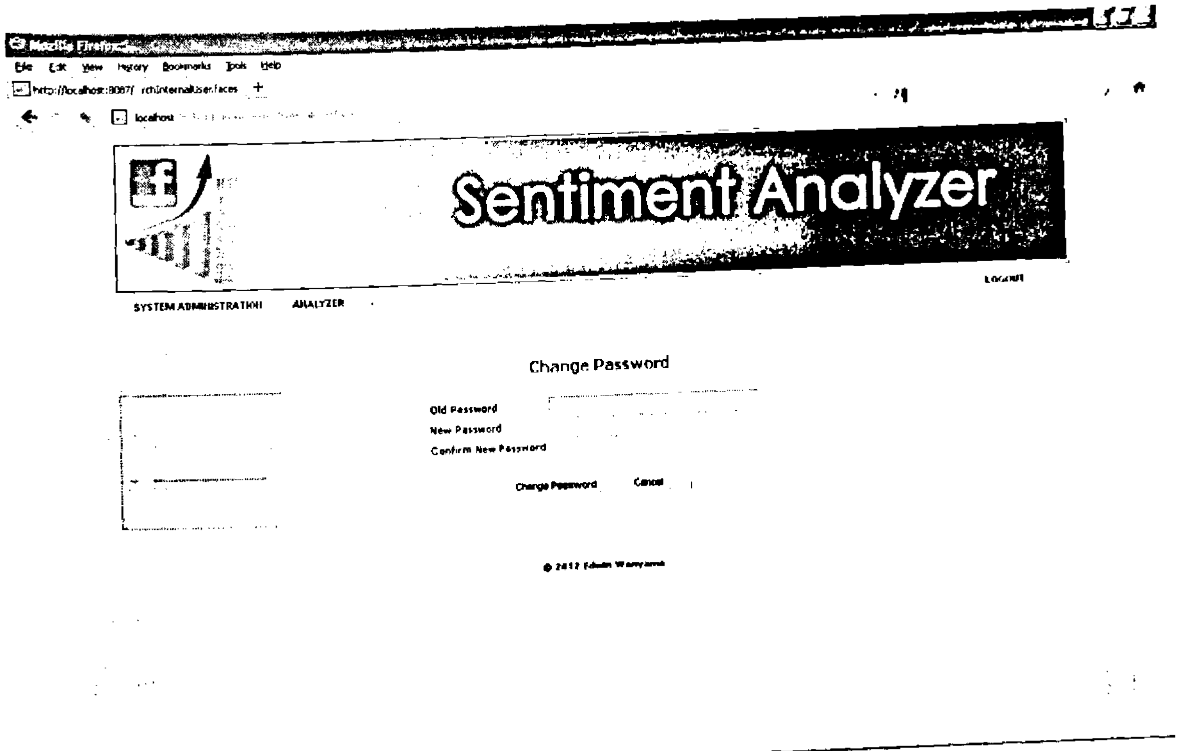
Choose	User Name	Staff No	Surname	Other Names	Email
	H10021C33	10021033	Wekesa	Devis	[illegible]@gmail.com

UI5 - Modify User Details Form

Change Password

To change your password, do the following:-

1. While logged in, navigate to the 'SYSTEM ADMINISTRATION' menu and select the 'CHANGE' password submenu.
2. System responds by displaying interface **UI6 – Change Password Form.**
3. Enter old password, new password and confirm the new password then click on 'Change Password' to effect the changes.
4. To exit the interface click on 'Cancel'.



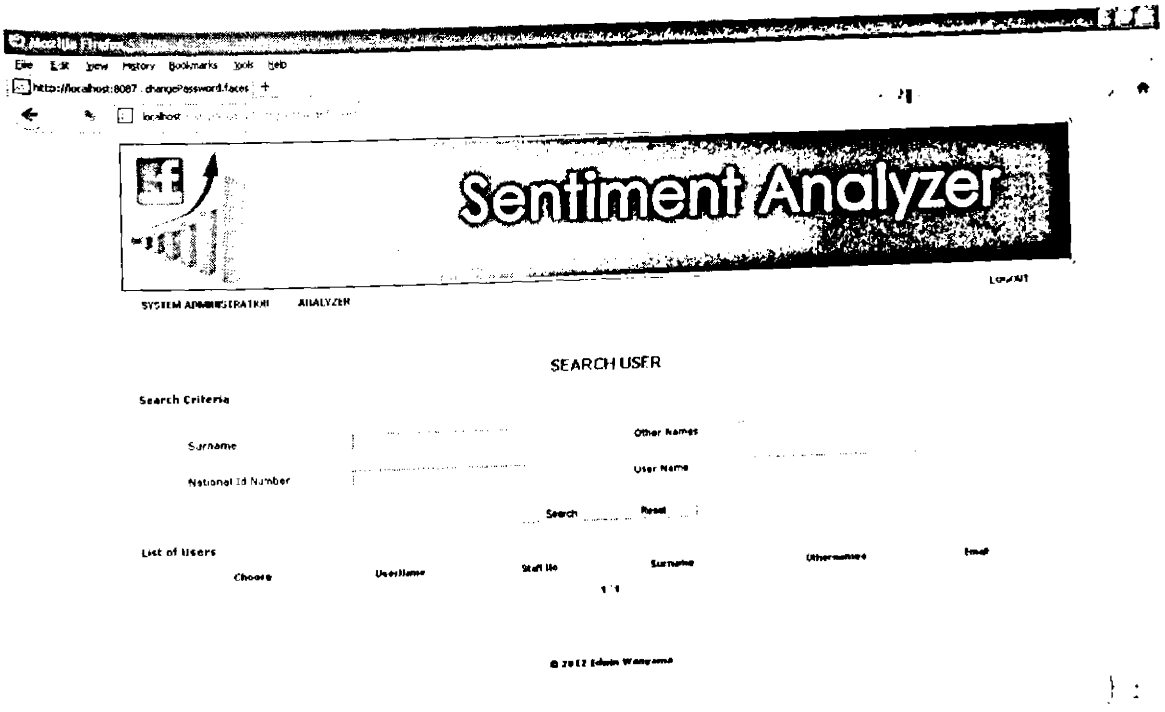
UI6 - Change Password Form

Search User

To search for a user existing in the system, do the following:-

1. While logged in the system, navigate to the 'SYSTEM ADMINISTRATION' menu and select the 'SEARCH USER' submenu.
2. System responds by displaying interface **UI7 – Search User Form.**
3. Enter search criteria then click 'Search' button to locate the user of interest.

4. System will respond with a list of users meeting search criteria provided otherwise no results will be returned.
5. Select the user record of interest by clicking on the 'Choose' icon on the extreme left column of the record.
6. System will respond with a display of the user details in read only mode.
7. View the details and click cancel button to exit.



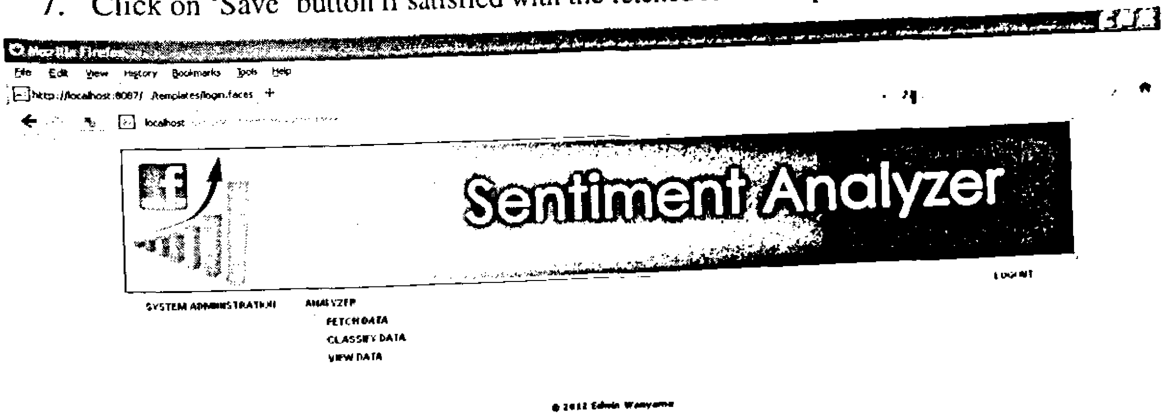
UI7 - Search User Form

Fetch Data from Facebook and Twitter

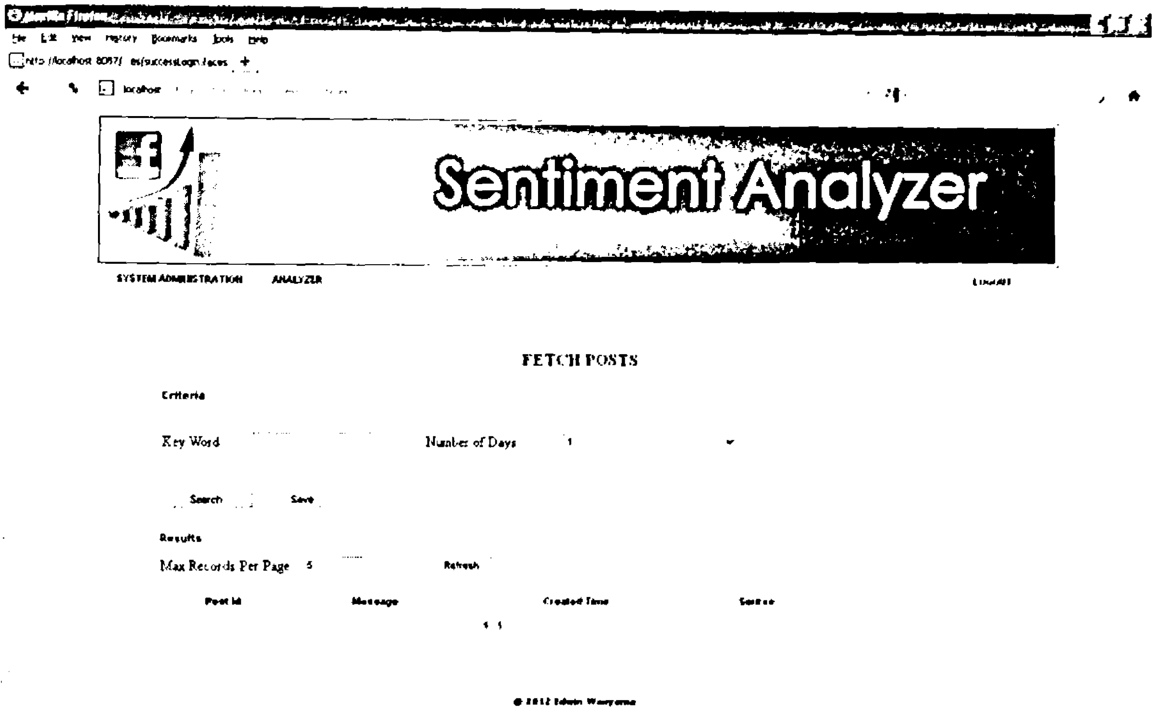
To fetch data from Facebook and Twitter perform the following steps:-

1. While logged in the system, navigate to the 'ANALYZER' menu and select the 'FETCH DATA' submenu as shown in interface **UI8 – Analyzer Menu and Menu Items**
2. System responds by displaying **UI9 – Fetch Data Form shown.**
3. Enter keyword(s) to search for and far back in terms of days that the search should go to from the current date.

4. Click on 'Search' button to fetch the data.
5. System will respond with a display of the fetched results if any data containing the keyword(s) is found.
6. View data displayed by paging through with the help of the scrolling tool that is beneath the records. You can also change the number of records displayed by changing the 'Max records' field then refresh to view more or less.
7. Click on 'Save' button if satisfied with the fetched results to persist the data.



UI8 - Analyzer Menu and Menu Items



FETCH POSTS

Criteria

Key Word

Number of Days 1

Results

Max Records Per Page 5

Post Id	Message	Created Time	Source

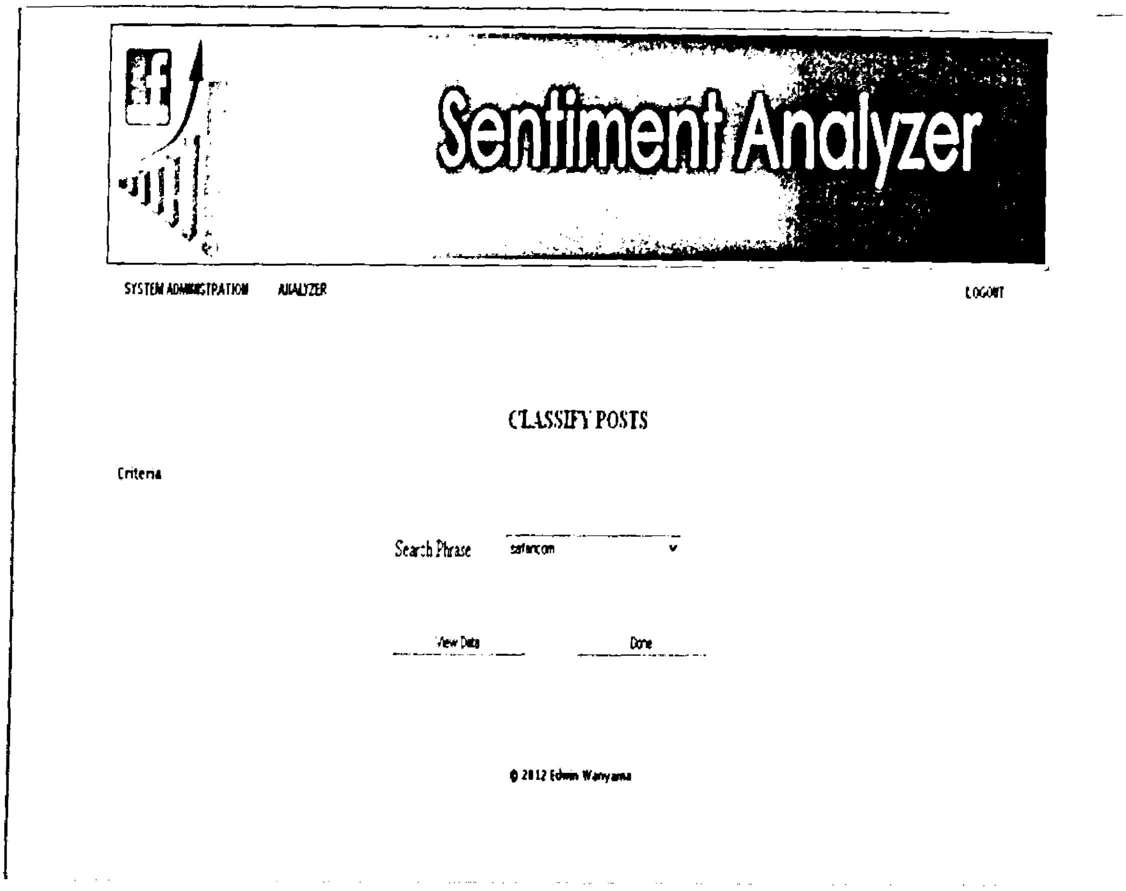
© 2012 Edwin Wanyama

UI9 - Fetch Data Form

Classify Data

To classify Data do the following steps:-

1. While logged in the system, navigate to the 'ANALYZER' menu and select the 'FETCH DATA' submenu as shown above in interface **UI8 – Analyzer Menu and Menu Items**
2. System responds by displaying **UI10 – Classify Data Form.**
3. Select messages that relate to a particular keyword by selecting the keyword using the dropdown field given on the interface.
4. Click on 'View' button to view the unclassified messages.
5. Click on 'Classify' button to classify the messages.
6. System informs you to wait as it performs the classification. It then notifies you upon completion of the classification.
7. Click on 'Cancel' button to exit from the interface.



CLASSIFY POSTS

Criteria

Search Phrase safaricom

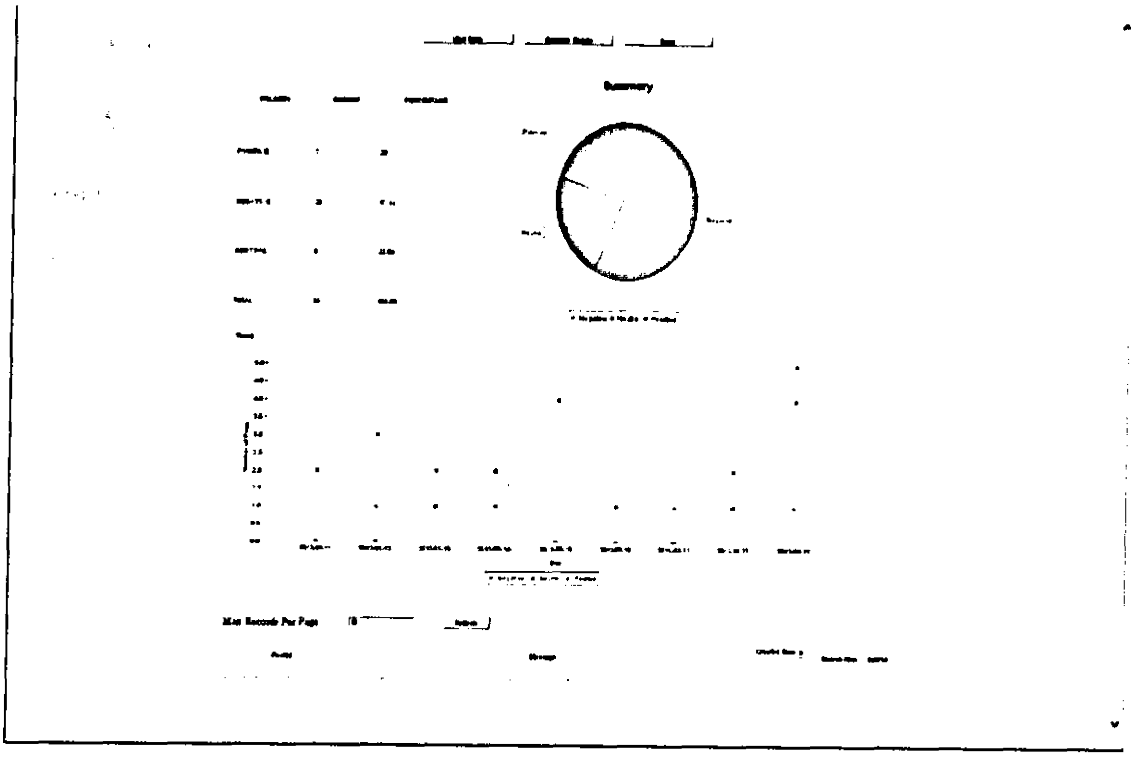
© 2012 Edwin Wanyama

UI10 – Classify Data Form

View Results

To view classification results, do the following:-

1. While logged in the system, navigate to the 'ANALYZER' menu and select the 'VIEW RESULTS' submenu as shown in interface **UI8 – Analyzer Menu and Menu Items**
2. System responds by displaying interface **UI11 – View Results Form**
3. Select 'Search Phrase' for which you want to view classification results then click on 'View Data' button.
4. System will display a tabulated summary of the classification results and also display the specific messages in colors that correspond to the three polarities (positive, negative and neutral). You can therefore view the summary as well as go to the specifics
5. Click on 'Generate Pie Chart' button to view a presentation of the classification summary in a Pie chart.
6. System will display a Pie Chart with three polarities coded in three different colors (Red color – representing Negative polarity, Blue color – representing Neutral polarity and Green color – representing Positive polarity).
7. Click on 'Generate Bar Chart' button to view the trends of the classification summary over time in days.
8. Click on 'Done' button to exit from the interface.



UI11 - View Results Form

Appendix 4 – Test Cases and Test Evidence

i. Authenticate User Module

Precondition: - Browser launched and application accessed by user through its URL

Test case id	Test case name	Test case description	Test steps		
			Steps	Expected	Actual/Outcome
TC-1	Aunthenticate user Test Case	To verify that the 'Authenticate user' functionality works correctly	Enter invalid login Id and valid password and click on 'Submit' button	System should give 'Invalid Username' message	System gives 'Invalid Username' message
			Enter valid login Id and invalid password and click on 'Submit' button	System should give 'Wrong User name or password' message	System gives 'Wrong User name or password' message
			Enter invalid login Id and invalid password and click on 'Submit' button	System should give 'Wrong User name or password' message	System gives 'Wrong User name or password' message
			Enter valid login Id and blank password and click on 'Submit' button	System should give 'Wrong User name or password' message	System gives 'Wrong User name or password' message

ii. Create user

Precondition: - User is successfully logged in and has the function for creating users accessible via the interface.

Test case id	Test case name	Test case description	Test steps		
			Steps	**Expected**	**Actual/Outcome**
TC-2	Create User Test Case	To verify that the 'Create user' functionality works correctly	Activate module by clicking on 'Create User' submenu	System should display form for capturing new user's details	System displays form for capturing user details
			Provide incomplete user details and click the 'Save' button	System should provide a list of the missing details in red color for user to see and rectify	System gives a list of the missing values in red color for clear visibility
			Enter invalid email address	System should give message 'The email must be in the format yourname@yourdomain.com'	System gives 'The email must be in the format yourname@yourdomain.com' message
			Click on 'Cancel' button	System should exit the user details interface and take focus back to the main form without saving details captured if any.	System exits the user details interface and takes focus back to the main form.
			Click on 'Save' button after capturing all required	System should save new user's details, create an account for	System saves new user's details, creates an account for the new user

			details	the new user then generate and display the login credentials for the user	then generates and displays the login credentials for user

iii. Modify User Details

Test case id	Test case name	Test case description	Test steps Steps	Expected	Actual/Outcome
TC-3	Modify User Test Case	To verify that the 'Modify User Details' functionality works correctly	Activate module by clicking on 'Modify User Details' submenu	System should give 'Search User' interface	System displays 'Search User' interface
			Click on 'Search' button without entering any search criteria	System should give a list of all users in the system	System gives a list of all users in the system
			Enter criteria that does not match details of any user existing in the system	System should fetch any results	No results displayed
			Choose a record from the displayed list by clicking on the icon under the 'Choose' column	System should display user's details on another interface in edit mode.	System displays user's details on another interface in edit mode
			Make changes to user record and click on 'Save' button	System should confirm successful save operation for the changes made	System confirms successful save operation for the changes made on the user

				on the user record.	record.
			Make changes to user record by removing some details like email and click 'Save' button	System should notify the user that the missing details are required	System does notify the user that the omitted details are required
			Click on 'Cancel' button while in the midst of making changes	System should exit the modify user details interface without saving changes that have been done	System exits the interface without saving changes

iv. Change Password

Test case id	Test case name	Test case description	Test steps		
			Steps	**Expected**	**Actual/Outcome**
TC-4	Change Password Test Case	To verify that the 'Change Password' functionality works correctly	Activate module by clicking on 'Change Password' submenu	System should display 'Change Password' interface	System displays 'Change Password' interface
			Enter old password, new password and confirm new password then click on 'Change Password' button	System should acknowledge successful change of password via "password changed successfully" message to the user.	System acknowledges successful change of password via "password changed successfully" message to the user.
			Enter new password that is less than 8 characters	System should give message 'Passwords must contain a	System gives message 'Passwords must contain a minimum of 8

				minimum of 8 characters and must contain at least one digit and character'	characters and must contain at least one digit and character'
			Enter new password that does not contain a digit	System should give message 'Passwords must contain a minimum of 8 characters and must contain at least one digit and character'	System gives message 'Passwords must contain a minimum of 8 characters and must contain at least one digit and character'
			Enter new password that does not contain an alphabetic letter	System should confirm successful save operation for the changes made on the user record.	System confirms successful save operation for the changes made on the user record.
			Enter new password and confirmation password that do not match	System should give message 'The New Password and Confirm Password do not match, please try again'	System gives message 'The New Password and Confirm Password do not match, please try again'
			Click on 'Cancel' button while in the midst of making changes	System should exit the modify user details interface without saving changes that have been done	System exits the interface without saving changes

v. **Fetch Data**

Test case id	**Test case name**	**Test case description**	**Test steps**		
			Steps	**Expected**	**Actual/Outcome**
TC-5	Fetch Data Test Case	To verify that the 'Fetch Data' functionality works as per the design	Activate module by clicking on 'Fetch Data' submenu which is under the Analyzer menu	System should display 'Fetch data Form' interface	System displays 'Fetch data Form' interface
			Enter keyword and number of days then click on 'Search' button	System should inform user to wait as it searches for posts that have the specified keyword(s). System displays results if matching data is found.	System informs user to wait as it searches for posts that have the specified keyword(s). System displays results if matching data is found.
			Click on 'Search' button without specifying keyword	System should ask user to provide search keyword	System demands for search keyword
			Change value of maximum records to be displayed per page and click on the refresh button.	System should adjust the number of records displayed per page to the new number specified if the records available are equal to or more than the	System adjusts the number of records displayed per page as required

				number specified	
			Click on 'Save' button once data has been fetched and considered to be satisfactory	System should confirm successful save operation for the data fetched.	System confirms successful save operation for the data fetched.

vi. Classify Data

Test case id	Test case name	Test case description	Test steps		
			Steps	Expected	Actual/Outcome
TC-6	Classify Data Test Case	To verify that the 'Classify Data' functionality works as per the design	Activate module by clicking on 'Classify Data' submenu which is under the Analyzer menu	System should display 'Classify Data Form' interface	System displays 'Classify Data Form' interface
			Choose keyword/search phrase from drop down field and click on 'Classify Data' button	System should inform user to wait as it classifies data related to the selected keyword then notify user of successful completion of process.	System classifies data and notifies user upon completion as expected.
			Select a Search phrase and click on 'View Data' button without specifying keyword	System should display a list of unclassified messages in read only format for the user to view	System does display unclassified messages in viewable form
			Click on	System	System does

			'View Data' button without selecting a search phrase	should notify user to specify a search phrase	notify user to specify a search phrase
			Click on 'Cancel' button	System should exit from the Classify data form	System does exit from the Classify data form

vii. View Results

Test case id	Test case name	Test case description	Test steps		
			Steps	**Expected**	**Actual/Outcome**
TC-7	View Results Test Case	To verify that the 'View Results' functionality works as per the design	Activate module by clicking on 'View Results' submenu which is under the Analyzer menu	System should display 'UI11 - View Results Form' interface	System displays 'UI11 - View Results Form' interface when the 'View Results' submenu is selected
			Click on View Data without specifying search phrase/ keyword	System should notify user to specify	System demands for a search phrase/ keyword
			Specify data to view by selecting search phrase related to it and click on 'View Data' button	System should display a tabulated summary of the classification results for the specified search phrase. It should also display all the individual messages in colors	System displays a tabulated summary of the classification results for the specified search phrase. It also displays all the individual messages in colors

				corresponding to their determined polarity.	corresponding to their determined polarity.
			Click on 'View Pie Chart' button when search phrase is still specified	System should display a summary of the classification for the chosen phrase in a pie chart.	System displays a pie chart showing classification summary in terms of percentage.
			Click on 'View Line graph' button when search phrase is still specified	System should display a Line graph of the classification for the chosen phrase for each polarity type.	System displays a Line graph of the classification for the chosen phrase. User can view trends for the three polarities
			Click on 'Cancel' button to exit the View Results Form	System should exit and take focus to Main Form when Cancel button is clicked	System exits and takes focus to Main Form when Cancel button is clicked