UNIVERSITY OF NAIROBI
SCHOOL OF COMPUTING AND INFORMATICS

A COMPARATIVE EVALUATION OF SENTIMENT ANALYSIS TECHNIQUES ON FACEBOOK DATA USING THREE MACHINE LEARNING ALGORITHMS: NAÏVE BAYES, MAXIMUM ENTROPY AND SUPPORT VECTOR MACHINES

ANYIM JULIANNE AKINYI

P58/76363/2012

SUPERVISOR

DR. ROBERT OBOKO

November, 2014

Project submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science of the University of Nairobi

DECLARATION

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

Signature:_____ Date:_____

Anyim Julianne Akinyi

P58/76363/2012

The Project has been submitted in partial fulfillment of the Requirements for the Degree of Master of Science in Computer Science at the University of Nairobi with my approval as the University Supervisor.

Signature:_____ Date:_____

Dr. Robert Oboko

School of Computing and Informatics

ACKNOWLEDGEMENT

I wish to sincerely thank the following for their contributions towards the successful completion of this project.

First, the **Almighty God** for the strength and good health He granted me throughout the duration of the project.

I am greatly indebted to my supervisor **Dr. Robert Oboko** for his good guidance and direction, advice, strong support, highly-valued criticisms, deep insights and for providing resourceful information that contributed to the quality of this project.

For praying for me and supporting all my endeavors, I give all my appreciation to my parents, **Mr. & Mrs. Anyim**.

Lastly, I wish to thank all my friends and colleagues for their interest in my work and ideas that contributed to the success of this project.

Anyim Julianne

ABSTRACT

The rapid growth and popularity of social networks has led to the creation of vast amounts of textual data often in an unstructured, fragmented and informal form. Huge volumes of electronic data in form of reviews, customer feedback, elicited surveys, unsolicited comments, suggestions and criticisms are generated on a daily basis which makes it difficult for institutions, government bodies, companies and prospective organizations to react to feedback quickly due to the inadequate capacity to handle the volumes.

While recent NLP-based sentiment analysis has centered around Twitter and product or service reviews, we believe it is possible to more accurately classify the emotion in Facebook status messages due to their nature. Facebook status messages are more concise than reviews and tweets, thus allowing for more characters to be used which means better writing and a more accurate portrayal of emotions.

In this study, we perform Sentiment Analysis on Facebook by fetching the posts and extracting their content. We then tokenize the data in order to extract their keyword combinations and perform feature selection to keep only the n-grams that are important for the classification problem. We finally train our classifier to identify the polarity of the posts i.e. whether positive, negative or neutral.

We analyze the suitability of various approaches to NLP sentiment analysis by comparing the performance of the Naïve Bayes Classifier, Maximum Entropy Classifier and Support Vector Machines. We notice that feature selection technique has a significant impact on the performance of the algorithm. The presence of trigram and bigram information produced better results with all the three algorithms compared to unigrams. This is attributed to the fact that trigrams and bigrams are better at capturing sentiment patterns unlike unigrams which just provide a good coverage of the data. Trigrams achieved an overally higher performance in all instances giving an accuracy of 82.6% with unigrams achieving the least accuracy of 73.8%. However, as statements became long and winded with contradictory phrases, the classifiers performed poorly. This means therefore, that feature selection method alone is not enough to determine the performance of an algorithm. Some advanced NLP techniques might be required to deal with this shortcoming.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

NLTK	Natural Language Tool Kit
NB	Naïve Bayes
ME	Maximum Entropy
SVM	Support Vector Machines
SNSs	Social Network Sites
NLP	Natural Language Processing
POS	Part of Speech
API	Application Programming Interface
HTML	Hyper Text Markup Language
CSS	Cascaded Style Sheets

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Background

The Internet provides an extensive array of services and applications, including news sites, trading services, entertainment and innumerable forums and chat rooms. Among the most popular web-based services are Social Network Sites (SNSs) such as Facebook, Twitter, Linked In, Flickr among others which are becoming increasingly popular as tools for online social interactions. The development and use of SNSs has revolutionized the way people share information and keep in touch with friends. People are able to express their opinions in form of posts (Facebook), tweets (Twitter), emoticons etc with regard to many issues that affect their day to day lives. According to Ipsos Synnovate Report (2009), 79% of internet users in Kenya are members of Facebook using it as a primary means to talk to friends, relatives, work-mates and to "follow" their favorite companies, organizations as well prominent personalities and politicians. The Twitter community, known as #KOT (KenyansOnTwitter) on the other hand, is particularly active, using this social media platform for online activism, praising corporate brands or calling them out and even rallying the public to help in cases of famine, protest campaigns and many others.

The Kenyan leadership has understood the power of social networks and has made it an integral part of their communication with the people. The president of Kenya Hon. Uhuru Kenyatta for example has a followership of close to a million on Facebook and over 400,000 on Twitter, and was recently ranked most followed president in Africa (Standard Digital, 2014). Wasswa (2013) in his research on the "The Role of Social Media in the 2013 Presidential Election Campaigns in Kenya" found that social media played a big role in the 2013 presidential elections with presidential candidates integrating social media into their campaigns. The platform was majorly used for sharing information on campaign activities, debate on issues, share photos, videos and links, solicit for funds, counter propaganda and update followers on the going ons.

The corporate world is not left behind with Safaricom, OLX Kenya, Airtel, Samsung Mobile Kenya, Equity Bank, Kenya Airways and Kenya Power among the leading embracers of social media with the largest followerships (Socialbaker, 2014).

These organizations use social media to generate conversations with their target groups, pertaining to their products and services and their activities and if done right often leads to an increase in traffic, brand buzz and meaningful sales. This wealth of social data is usually in an unorganized, fragmented and informal form making it difficult to get useful information from the sites as users have to spend a lot of time manually filtering the data and sometimes end up not getting the intended message. This problem creates an opportunity for improved information mining via NLP sentiment analysis.

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. Unlike Twitter which is limited to 140 characters, Facebook status messages can take well over 60,000 characters (Cohen, 2011) allowing for better writing and a more accurate portrayal of emotions.

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, statistics, 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. This study seeks to assess how various machine learning algorithms for sentiment analysis would perform on the very noisy domain of customer feedback data and propose the most appropriate based on the results observed. This would assist future developers who intend to integrate sentiment classifiers in their applications to know which algorithms would produce the best results.

1.2 Problem Statement

The inherent popularity and perceived value of social networks has led to the creation of vast amounts of textual data often in an unstructured, fragmented and informal form. Companies, governments and so on typically receive high volumes of electronic public feedback every day, some of it in the form of elicited surveys, some in the form of unsolicited comments, suggestions and criticism. The large number of reviews makes it difficult for companies or any government institution to react to feedback quickly and to direct it to the appropriate channels within the company / organization for action.

Further, readers or potential clients of these companies may not be able to quickly make informed decisions based on the unstructured data thus may give up as it may take too long going through the data in order to filter the information required to make a decision. To address the above challenges, it is desirable to provide an intelligent and automated system that can effectively organize and classify social media data so that it can be leveraged by human users in a meaningful way.

This would in effect provide valuable information ranging from rates of customer satisfaction to public opinion trends to policy-makers which can help them save money and improve customer satisfaction. Further, extracting the sentiment of a review can help provide concise summaries to readers and automatically generate useful recommendations for them.

While much work has recently focused on the analysis of social media in order to get a feel for what people think about current topics of interest, there are, however, still many challenges to be faced. Among such challenges is the inherent difficulty in extracting a singleton sentiment label from long passages, where fluctuations over the document length make classification difficult (Ssoriajr, Kanej 2010). Also, given the informal and unstructured nature of social media data it is necessary to perform an analysis of the various machine learning approaches for sentiment analysis to see which approach performs best or is the most desirable for the kind of data available on social networks.

This research tackles the problem of classifying Facebook posts based on the attitudes expressed in them (positive, negative or neutral) and generating summaries and trends of the classified data. These summaries can find applicability in various areas e.g. helping customer care service centers in maintaining better customer relations or helping government institutions in analyzing and responding quickly to public concerns or public figures and politicians to get a general view of the supporters' opinion and take the necessary action.

1.3 Research Objectives

The objectives of this study therefore are to:

- Develop a technique for harvesting and storing Facebook data.
- Analyze machine learning algorithms used in other classification models and evaluate their suitability for the problem of classifying Facebook posts for sentiment analysis.
- Develop classifiers using the algorithms identified above and extract features that will allow them to classify sentiments into the positive, negative or neutral.
- Conduct a comparative analysis of the performance of the algorithms .

1.4 Research Questions

This study will attempt to address the following questions:

- What are the issues surrounding harvesting of data from Facebook? Are there suitable cost effective techniques for this task?
- What features or characteristics of the Facebook environment are suitable for this study?
- What classification algorithms have been successfully used before in similar research tasks and what are some of the issues that surround the usage of these models?
- What is the best way to integrate the algorithms learnt into an application that can provide Facebook sentiment analysis from the point of data collection to presentation of results?
- Of the three NLP-based classification algorithms, which one produces the best results?

1.5 Research Justification

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

- A sentiment analyzer and classifier that can incorporates the Kenyan languages English and Kiswahili hence can be used in the Kenyan set up.
- A tool that can assist companies, government or individuals to organize feedback or review data from the populace in a more useful and meaningful way thus helping in improving service delivery and customer satisfaction.
- A tool that will provide an analysis of the various classification approaches such as Naïve Bayes, Maximum Entropy and Support Vector Machines to help determine the best classification approach for Facebook /Social media data.

CHAPTER 2: LITERATURE REVIEW

2.1 Meaning of Social Media

Social Media is an umbrella term that describes websites and online tools that people use to connect and share content, experiences, opinions and media (Amway, 2013). It enables conversations and interactions with people online. Examples of Social Media platforms are Facebook, Twitter and YouTube.

While social media is great for staying in touch with friends and family, it also provides businesses of all shapes and sizes with a fantastic opportunity to communicate directly with new and existing customers - and at minimal cost. Reaching hundreds, even thousands of existing or potential customers with up-to-date information and promotions has never been easier or cheaper. Both Facebook and Twitter allow small businesses to share descriptions about themselves, photographs, and information about their products and how to buy them, with new and existing customers at the click of a mouse.

2.2 The Growth of Social Media in Africa

In the mid-1990s, as the use of mobile phones started its rapid spread in much of the developed world, few thought of Africa as a potential market. Now, with more than 400 million subscribers, its market is larger than North America's. Africa took the lead in the global shift from fixed to mobile telephones as noted by the UN International Telecommunications Union (UN AfricaRenewal magazine, 2012).

Studies suggest that when Africans go online (predominantly with their mobile phones) they spend much of their time on social media platforms (Facebook, Twitter, YouTube and so on). Sending and reading e-mails, reading news and posting research queries have become less important activities for Africans.

In recent months Facebook, the major social media platform worldwide and currently the most visited website in most of Africa has seen massive growth on the continent. The number of African Facebook users now stands at over 17 million, up from 10 million in 2009. More than 15 per cent of people online in Africa are currently using the platform,

compared to 11 per cent in Asia. Two other social networking websites, Twitter and YouTube, rank among the most visited websites in most African countries.

Along with regular citizens, African stars, thinkers, political leaders and companies have rapidly joined the global conversation. The Facebook fan base of Côte d'Ivoire's football star and UN goodwill ambassador Didier Drogba is more than 1 million people. Zambian best-selling author and economist Dambisa Moyo has more than 26,000 followers on Twitter. Media organizations in South Africa and companies such as Kenya Airways are using various social media platforms to interact better with customers and readers. During recent elections in Côte d'Ivoire candidates did not only tour cities and villages; they also moved the contest online, feverishly posting campaign updates on Twitter and Facebook.

2.3 About Facebook

Facebook is an online social networking service consisting of a series of interrelated profile pages in which members post a broad range of information about themselves and link their own profile to others' profiles (Wilson et al., 2012). Facebook allows users to create their personal profiles and add other users as "friends". It also facilitates communication via its "message" system that allows for private communication and a "wall" system that allows for a more public form of communication. In addition, users may join common-interest groups organized by work place, school or college, and also categorize their friends into custom lists such as "work mates", "family", "close friends" etc. Facebook's "News Feed" feature allows users to receive automatic notifications on their profiles every time their friends update their profiles or engage in any form of online activity e.g. status updates and sorts these events in a chronological order.

A research by Ipsos Synnovate (2009) indicates that there were over 2 million users of Facebook in Kenya in 2009, using it as a primary means to talk to friends, relatives and work-mates, with a vast majority accessing the site from their mobile phones. This constituted 79% of Kenya's internet users. A similar research by Socialbakers (2012) indicates that Facebook penetration in Kenya in October 2012 was at 4.88% compared to the country's population and 48.86% in relation to number of internet users. The total number of users in Kenya had reached 1,952,380and had grown by more than 638,980 in the

6 months preceding October 2012.The user age distribution on Facebook in Kenya indicates that the largest age group is currently 18 - 24, followed by the users in the age of 25 – 34. This is depicted in Figure 1 below. The male to female ration is 67% to 33% respectively as depicted in Figure 2.

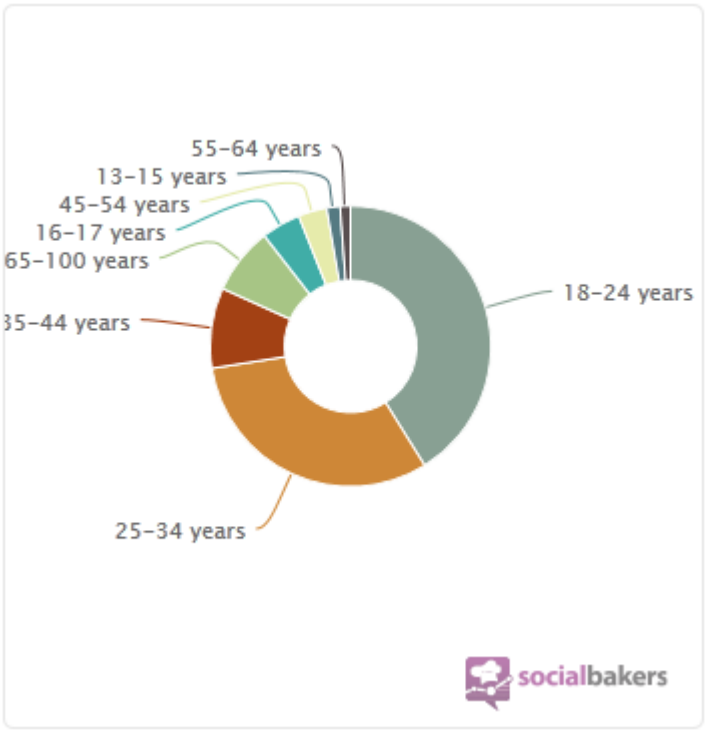

Figure 1: User age distribution of Facebook in Kenya (Source – Socialbakers 2014)

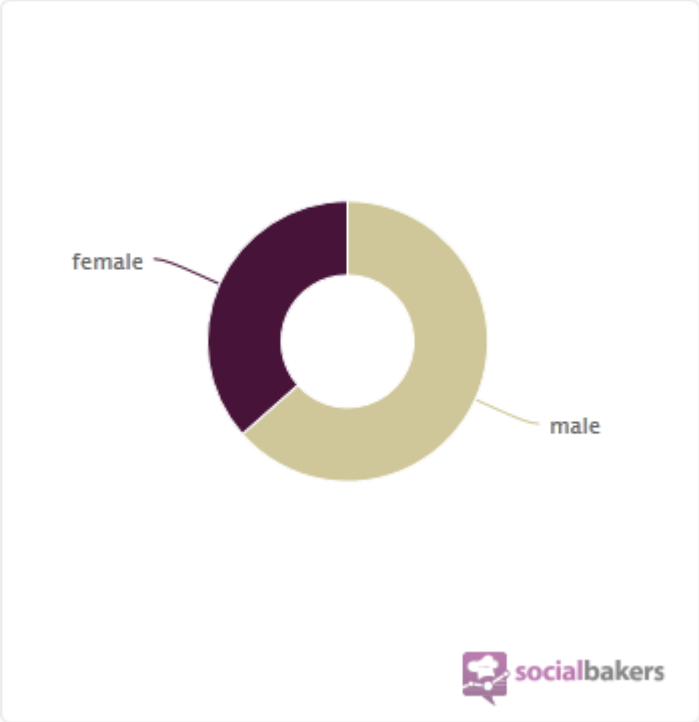

Figure 2: Male/Female user ratio on Facebook in Kenya (Source – Socialbakers 2014)

2.4 Machine Learning Algorithms

Machine learning is a scientific discipline that deals with the construction and study of algorithms that can learn from data (Kovahi and Provost, 1998). Such algorithms operate by building a model based on inputs (Bishop, 2006) and using that to make predictions or decisions, rather than following only explicitly programmed instructions.

Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning "signal" or "feedback" available to a learning system. These are:

- **Supervised learning** - The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- **Unsupervised learning** - No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end.

- **Reinforcement learning** - A computer program interacts with a dynamic environment in which it must perform a certain goal, without a teacher explicitly telling it whether it has come close to its goal or not. Another example is learning to play a game by playing against an opponent.

Among the most common Machine Learning approaches are below:

a) Decision Tree Learning

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value.

b) Artificial Neural Networks

An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is inspired by the structure and functional aspects of biological neural networks. Computations are structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

c) Bayesian Networks

A Bayesian network, belief network or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning.

d) Clustering

Cluster analysis is the assignment of a set of observations into subsets (called *clusters*) so that observations within the same cluster are similar according to some pre-designated

criterion or criteria, while observations drawn from different clusters are dissimilar. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis.

e) Classification

Classification is the task of assigning a label to an input. The general class of classification problems includes many kinds of input besides text. To cite a few well-known examples, there are market basket systems, anomaly detection systems, and vision systems. A market basket system tries to detect buying patterns from a buyer's purchasing records and other information about them and their friends; an anomaly detection system tries to detect deviation from normal event patterns that signal trouble, for example in a computer security setting, or in a credit card fraud detection setting. A vision system is a system that recognizes objects or events.

f) Genetic Algorithms

A genetic algorithm (GA) is a search heuristic that mimics the process of natural selection, and uses methods such as mutation and crossover to generate new genotype in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms found some uses in the 1980s and 1990s

2.5 Classification

As described in (e) above, classification is the task of assigning a label to an input. There are two basic steps to using a classifier: training and classification. Training is the iterative process of taking content that is known to belong to specified classes and creating a classifier on the basis of that known content. Classification is a one-time process of taking a classifier built with such a training content set and running it on unknown content to determine class membership for the unknown content.

There are two main approaches for classification: supervised and unsupervised classification. 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).

Text classification is a special kind of classification problem. There are many practical applications of text classification. Below are some of the best-known examples:

a. **Spam filtering:** - a process which tries to discern E-mail spam messages from legitimate emails.

b. **Language guessing:** - automatically determining the language of a text

c. **Email routing:** - sending an email sent to a general address to a specific address or mailbox depending on topic.

d. **Sentiment analysis:** - determining the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document.

e. **Categorizing news feed topics:** – classifying text according to topics.

A generic text classification system is shown in the figure 3 below:

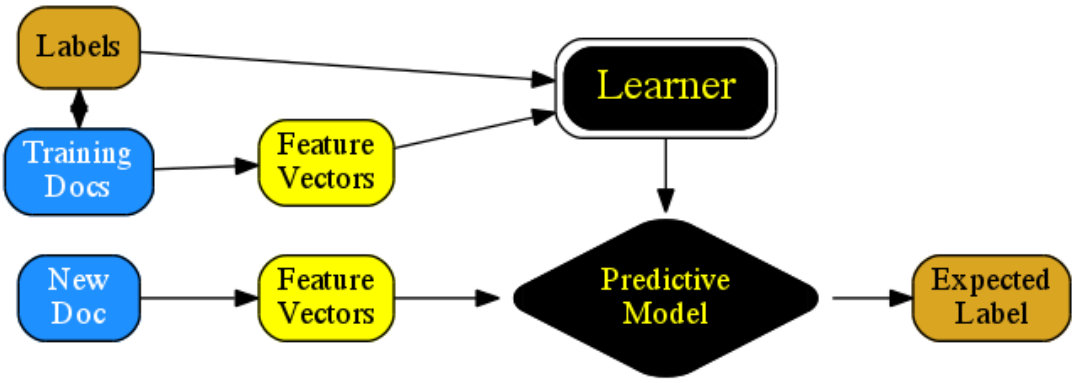

Figure 3: Generic text classification system

First there is a set of labeled training documents, which means we have, not just the documents, but some indication of what class they belong to from a small set of classes. From each document a vector of features is extracted. The features are the representations of the documents which the learner uses to try to draw generalizations about how to predict

classes. From the feature representations of the training documents and their labels, the learner produces a classifier. This phase is called the training phase. The classifier produced in the training phase can be used to classify new, unseen documents. To do this, features are extracted from the new document; the features are passed to classifier, and a classification decision (expected label) is produced.

2.6 Text Classification Techniques.

The most common Machine Learning algorithms for sentiment classification are Naïve Bayes, Maximum Entropy and Support Vector Machine. These are described below.

2.6.1 Naïve Bayes Classifier

The Naive Bayes classifier is an extremely simple classifier that relies on Bayesian probability and the assumption that feature probabilities are independent of one another (Vachaspati, P and Wu, C., 2012). In simple terms, a naive bayes classifier assumes that the value of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. This assumption is called *class conditional independence*. For example, a fruit may be considered to be an apple if it is red, round, and about 3" in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of the presence or absence of the other features.

Naïve Bayes classifiers have also exhibited high accuracy and speed when applied to large database. While applying Naïve Bayes classifier to classify text, each word position in a document is defined as an attribute and the value of that attribute to be the word found in that position.

Naïve Bayes is formalized as the product of the prior probability which is based on previous experience and the likelihood of a given attribute being in a given class, this forms the posterior probability.

To classify an unlabeled example it is just a matter of using the prior probabilities of a given category and multiplying them together. The category which produced the highest probability would be the label/classification for the unlabeled example. Only the words found in the unlabeled example would be looked up in the feature vector.

The following equation would be used to classify an unlabeled example. Given a document d and a class c, where the goal is to predict the probability that the document d belongs to class c.

$$P(c/d) = \text{argmax}\ (P(d/c).P(c))$$

Naïve Bayes is used when you have limited resources in terms of CPU and Memory. Moreover when the training time is a crucial factor, Naive Bayes comes handy since it can be trained very quickly.

2.6.2 The Maximum Entropy Classifier

The Max Entropy classifier is a probabilistic classifier which belongs to the class of exponential models. Unlike the Naive Bayes classifier discussed in the previous section, the Max Entropy does not assume that the features are conditionally independent of each other. The MaxEnt is based on the Principle of Maximum Entropy and from all the models that fit the training data, selects the one which has the largest entropy. The Max Entropy classifier can be used to solve a large variety of text classification problems such as language detection, topic classification, sentiment analysis and more.

Due to the minimum assumptions that the Maximum Entropy classifier makes, it is regularly used when nothing is known about the prior distributions and when it is unsafe to make any such assumptions. Moreover Maximum Entropy classifier is used when the conditional independence of the features cannot be assumed. This is particularly true in Text Classification problems where the features are usually words which obviously are not independent. The Max Entropy requires more time to train compared to Naive Bayes, primarily due to the optimization problem that needs to be solved in order to estimate the parameters of the model. Nevertheless, after computing these parameters, the method provides robust results and it is competitive in terms of CPU and memory consumption.

2.6.3 The Support Vector Machine Classifier

Support Vector Machines (SVMs) operate by separating points in a d-dimensional space using a (d-1)-dimensional hyperplane, unlike Max-Ent and Naive Bayes classifiers, which use probabilistic measures to classify points. Given a set of training data, the SVM classifier finds a hyperplane with the largest possible margin; that is, it tries finds the hyperplane such

that each training point is correctly classified and the hyperplane is as far as possible from the points closest to it. In practice, it is usually not possible to find a hyperplane that separates the classes perfectly, so points are permitted to be inside the margin or on the wrong side of the hyperplane. Any point on or inside the margin is referred to as a support vector, and the hyperplane, given by

$$f(\vec{B}, B_0) = \{\vec{x}^{|} \cdot \vec{B} + B_0 = 0\}$$

is selected through a constrained quadratic optimization to minimize

$$\frac{1}{2}|\vec{B}|^2 + C\sum_i \zeta_i$$

given

$$\forall i, y_i(\vec{x_i}^T \cdot \vec{B} + B0) \geq 1 - \zeta_i$$

2.7 The Concept of Sentiment Analysis

Sentiment analysis/classification (or opinion mining) is defined as the task of finding the opinions of authors about specific entities. The decision-making process of people is affected by the opinions formed by thought leaders and ordinary people. When a person wants to buy a product online he or she will typically start by searching for reviews and opinions written by other people on the various offerings. Sentiment analysis is one of the hottest research areas in computer science. Over 7,000 articles have been written on the topic. Hundreds of startups are developing sentiment analysis solutions and major statistical packages such as SAS and SPSS include dedicated sentiment analysis modules. There is a huge explosion today of 'sentiments' available from social media including Twitter, Facebook, message boards, blogs and user forums. These snippets of text are a gold mine for companies and individuals that want to monitor their reputation and get timely feedback about their products and actions. Sentiment analysis offers these organizations the ability to monitor the different social media sites in real time and act accordingly. Marketing managers, PR firms, campaign managers, politicians and even equity investors and online shoppers are the direct beneficiaries of sentiment analysis technology (Feldman, 2013).

2.8 Sentiment Analysis Feature Selection / Extraction

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 co-ordinate 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.

There are four feature categories that have been used in previous sentiment analysis studies. These include syntactic, semantic, link-based, and stylistic features (Abbasi A., Chen H. and Salem A., 2007). Along with semantic features, syntactic attributes are the most commonly used set of features for semantic analysis. These include word n-grams (Pang et al., 2002; Gamon, 2004), part of speech (POS) tags (Pang et al., 2002 and Yi et al., 2003), and punctuation.

2.9 Previous Sentiment Analysis 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).

Ngero E. W. and Wagacha P. W (2013) conducted sentiment analysis on product reviews on Facebook and Twitter using the Naïve Bayes classifier approach. The results they obtained with the classifier they developed was able to achieve an accuracy of 78.9%, a near human accuracy, as apparently people agree on sentiment only around 80% of the time. In their discussion they concluded that the Naive Bayes technique is ideal for the kind of data collected from social media on Kenyan products and services. As part of future research they suggested an assessment of other classifier algorithms such as Decision trees and

Support Vector Machines and a comparison made with the Naïve Bayes to see which classifier best suits the kind of data available on social media.

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.

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.

Pang et al., (2002) conducted sentiment analysis on movie review data using three supervised sentiment classification methods Naive Bayes, maximum entropy classification, and support vector machines. In their experiments they found that standard machine learning techniques definitively outperform human-produced baselines. They concluded that the three sentiment classification techniques above do not perform as well on sentiment classification as on traditional topic-based categorization.

CHAPTER 3: METHODOLOGY

3.1 Introduction

This chapter presents the system development methodology that was employed in conducting the study. We implemented the following three supervised machine learning methods for sentiment classification: the Naïve Bayes, Maximum Entropy and Support Vector Machines and at the end evaluated how each one of them performs in classifying Facebook sentiments using a pre-defined evaluation criteria.

3.2 System Development Methodology

The system development methodology chosen for this study was the Agile development methodology. Agile development methodology provides opportunities to assess the direction of a project throughout the development lifecycle. The methodology is described as “iterative” and “incremental” in that every aspect of development - requirements, design e.t.c. is continually revisited throughout the lifecycle. Agile development methodology is chosen due its following merits:

- Reduced development costs as the requirements are often revisited earlier in the development process enabling one to fine-tune the requirements before it is too late.
- Easy to adapt to changes and uncertainty.
- Working software is delivered much faster than in other methodologies like the waterfall model.

A generic Agile development process is shown below:

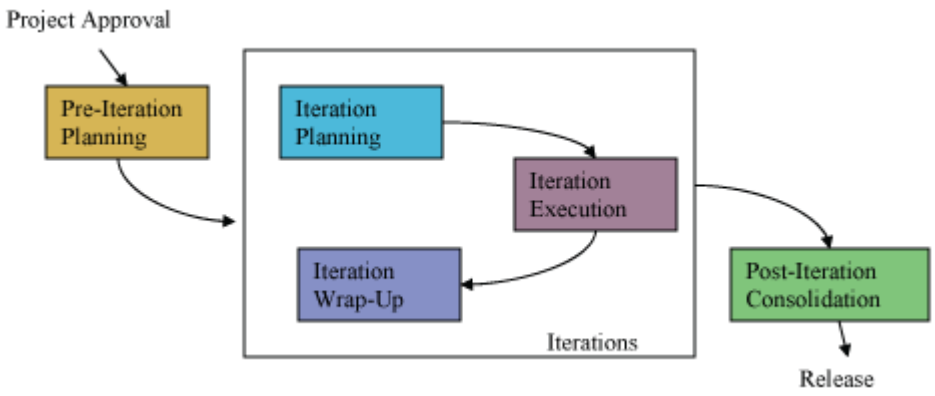

Figure 4: Agile development process

Every iteration goes through the Requirements Analysis, Design, Development and Testing, Integration and validation.

Agile Software Development									
Planning									
Analysis	Deploy	Analysis	Deploy	Analysis	Deploy	Analysis	Deploy	Analysis	Deploy
Develop		Develop		Develop		Develop		Develop	
Test		Test		Test		Test		Test	
Integrate		Integrate		Integrate		Integrate		Integrate	
Validate		Validate		Validate		Validate		Validate	

3.2.1 Requirements Analysis

The process of requirements gathering and analysis for developing the prototype will involve evaluating current sentiment analysis

3.2.2 Architectural System Design

In order to build the Facebook Sentiment Analysis tool, the following was done:

i. Extraction of posts from Facebook using an extraction script. We used the Facebook Graph API to collect posts and comments from Facebook. The posts were then stored in a database for the purposes of processing and analysis.

ii. Processing/Cleaning of the data to remove any irrelevant features which might interfere with the performance of the classifier. Cleaning inlvolved removing irrelevant characters and unnecessary repetitions so that the classifier is trained on clean data thus ensuring optimum performance.

iii. The data was then divided into training and test sets. 75% of the data was used for training while 25% for testing.

iv. Training the classifiers in order to come up with a model that be used to classify future posts.

v. Using the models to classify posts, a process that will extract features from the comments collected and classify them into positive, negative and neutral.

vi. The final step is to analyze the results obtained from the classifiers developed and draw conclusions and suggestions.

The above process is illustrated in Figure 5 below:

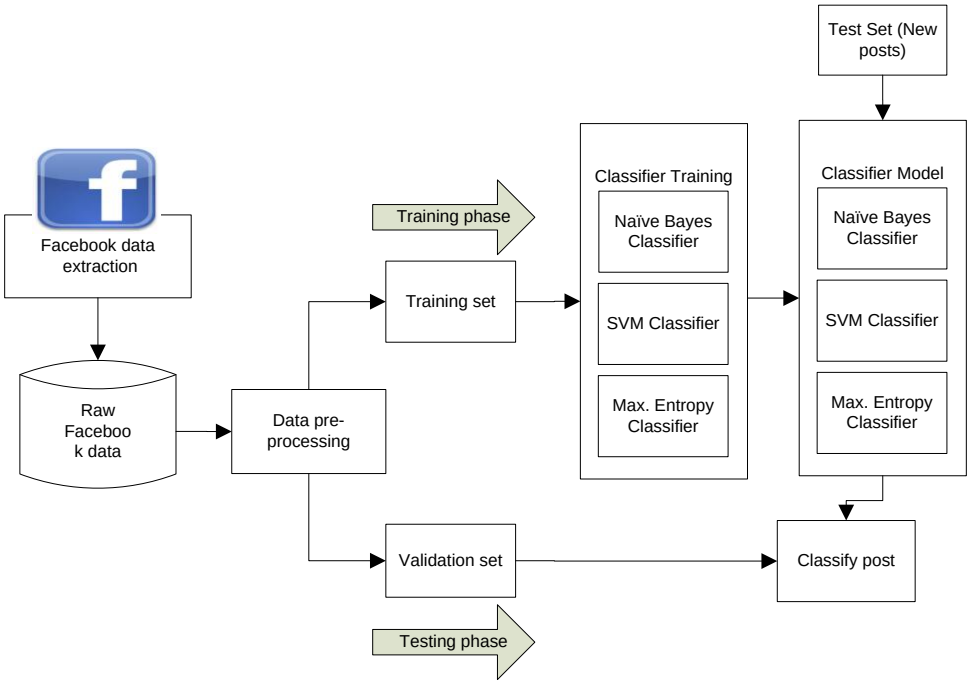

Figure 5: High level view of the classifier development approach

3.2.3 Classifier Development

The classifiers were developed using the Python Natural Language Toolkit (NLKT). NLTK is a leading platform for building Python programs to work with human language data. It

provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing and semantic reasoning (NLTK 3.0 documentation, 2014).

3.2.4 Analysis of the classification models

To evaluate the performance of the classification models developed, we used the NLTK metrics module which provides inbuilt functions for calculating accuracy, precision and recall for the classifier.

Precision measures the exactness of a classifier. A higher precision means less false positives, while a lower precision means more false positives. This is often at odds with recall, as an easy way to improve precision is to decrease recall.

Recall measures the completeness, or sensitivity, of a classifier. Higher recall means less false negatives, while lower recall means more false negatives. Improving recall can often decrease precision because it gets increasingly harder to be precise as the sample space increases.

3.2.5 Validation of the Prototype

The aim of this study as previously mentioned was to evaluate how the three sentiment analysis techniques perform in classifying Facebook data. To this end we developed an application that integrates the three techniques and performs an analysis of the performance of each. The system fetches posts and stores them in a database, preprocesses the data and trains the classifiers on a set of predefined features. It then uses the developed models to classify the sentiments expressed in new posts. To evaluate the application for completeness or in terms meeting the objectives of the study, the following criteria were used. An affirmative response to each of the questions confirmed the success of the study.

i. Is the tool able to collect data from Facebook?
ii. Is the classifier able to be trained with the data collected?
iii. Are the features selected for classification and training ideal?
iv. Are the results of the tests on the data accurate or are the results unfavorably skewed towards one sentiment?

v. Is the application developed able to provide a visual analysis of the performance of Naïve Bayes, Maximum Entropy and Support Vector Machines?

Table 1 below is a summary of the activities that were carried out to achieve the objectives of this study.

No	Objective	How objective will be achieved
1.	Develop a technique for harvesting and storing Facebook data.	The Facebook Graph API was customized and used to extract data from Facebook. The data was then stored in a database from where it was cleaned and stored in tables labeled positive, negative and neutral depending on the polarity of the message.
2	Analyze machine learning models used in other classification models and evaluate their suitability for the research problem.	Through literature survey of machine learning models for sentiment classification
3	Develop classifiers using the techniques identified above and extract features that will allow them to classify sentiments into the positive, negative or neutral.	Using the Python Natural Language Toolkit.
4	Conduct a comparative analysis of the performance of the techniques employed and make conclusions.	Using the NLTK metrics module for calculating accuracy, precision and recall.

Table 1: Summary of Project Activities

CHAPTER 4: SYSTEM ANALYSIS, DESIGN AND IMPLEMENTATION

4.1 Introduction

This chapter describes the preliminary analysis of the system to ascertain the detailed requirements in order to achieve the main objectives of the system, the system design and implementation. Under Systems Analysis we detail the functional and non-functional requirements of the system. Under Design we show the architectural design of the system, the flow of data processing, the database design and use case models.

4.2 Systems Analysis

4.2.1 Functional Requirements

In order to meet the objectives of the study, the application developed should be capable of doing the following:

i. Extract posts from Facebook and store them in a database for purposes of processing and analysis.

ii. Process the data to remove the low information gain features.

iii. Train the Naïve Bayes, Maximum Entropy and Support Vector Machine classifiers to come up with a model that can be used to classify new posts.

iv. Classify posts using the models developed by extracting the relevant features into positive, negative and neutral.

v. Produce a visual analysis on the results obtained.

4.2.2 Non-Functional Requirements

i. An easy to use and navigate Graphical User Interface.

ii. Simple and easy to interpret graphs.

4.3 System Design

4.3.1 Architectural System Design

This depicts from a broader perspective how the various components of the system are interlinked. The system was designed in a 4-tier architecture consisting of the Data Layer, the Data Access Layer, the Business Logic Layer and the User Interface Layer as shown in Figure 6:

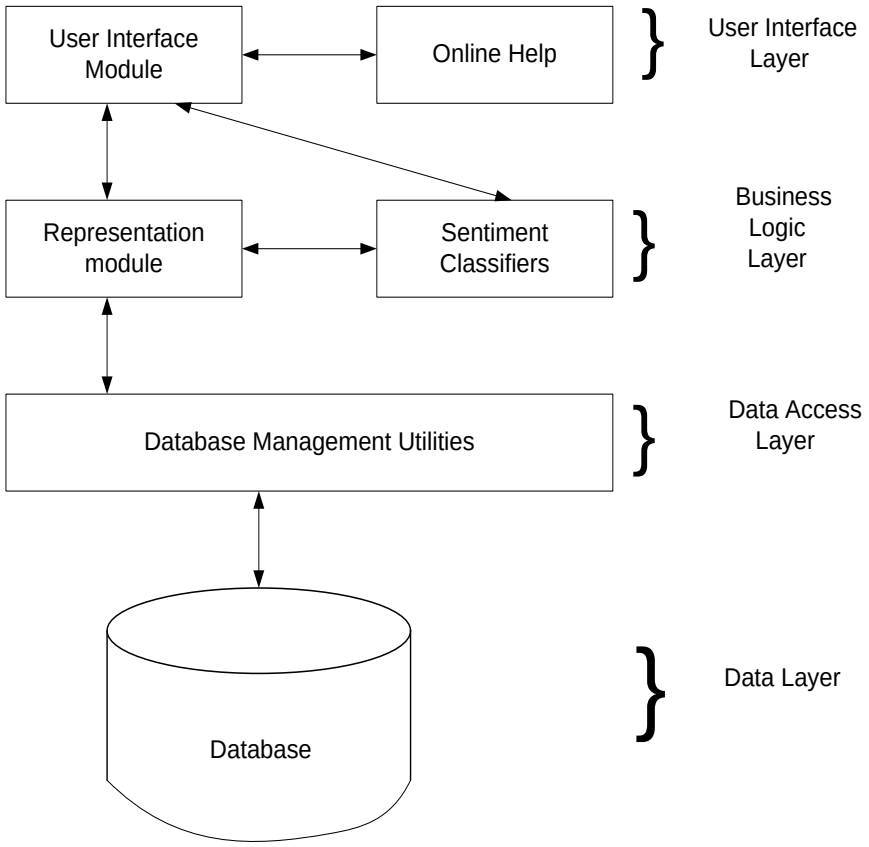

Figure 6: Overall System Architecture

- The **Data Layer** stores posts extracted from Facebook.
- The **Data Access Layer** serves retrieval requests from the upper layers.
- The **Business Logic Layer** can be thought of as the main system engine. It has the working logic for the system. It contains implementations for the data structure representations and the classifier models.
- The **User Interface Layer** enables a user to interact with the system. It carries out user commands and presents results generated by the system to the user.

4.3.2 Use Case Diagram

A use case is a complete sequence of related actions initiated by an actor. An actor is an external entity that interacts with the system. Use cases aid in the understanding of the functional requirements of a system.

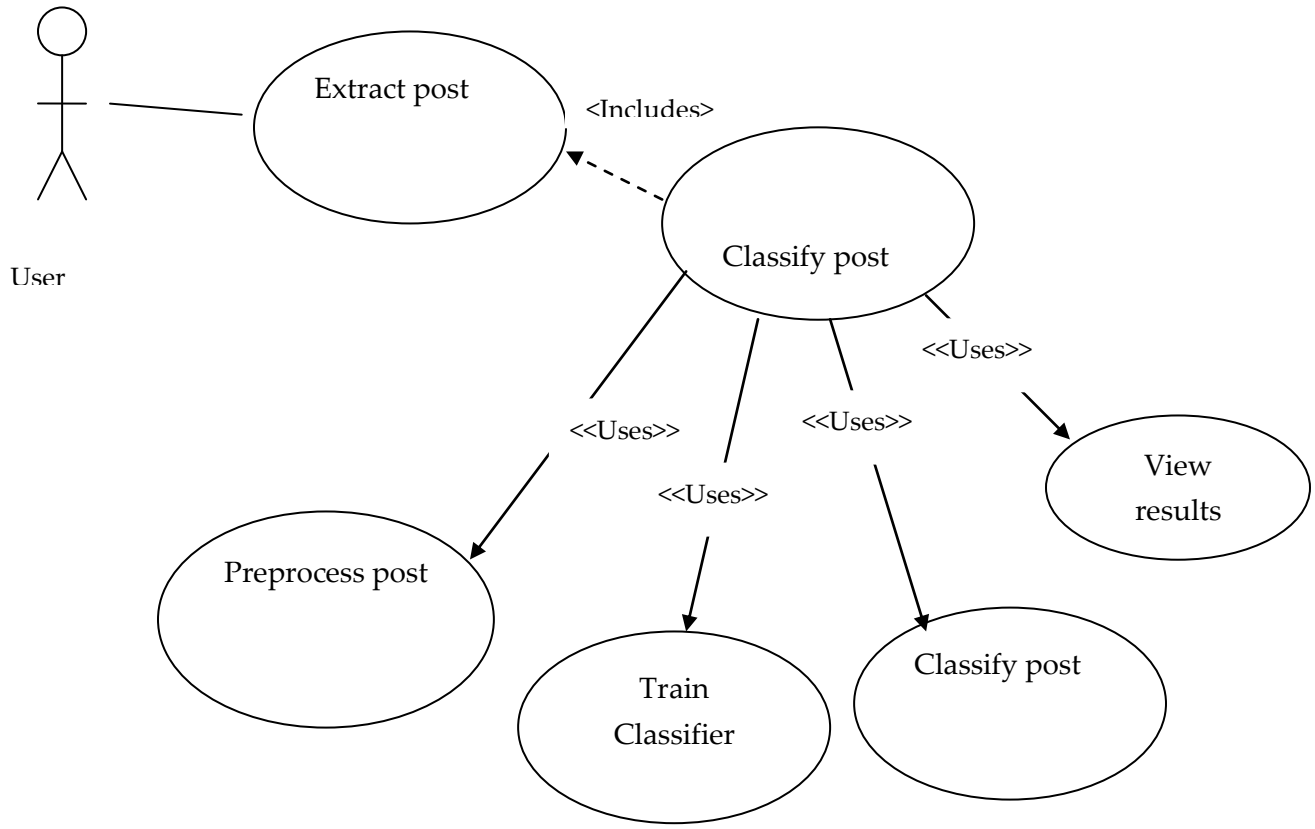

Figure 7: Use case diagram for the system (Source: Author)

4.3.3 Sentiment Analyzer Architectural Design

Figure 8 depicts an overview of the sentiment analyzer system components and the interconnections. The various modules are described in detail further below.

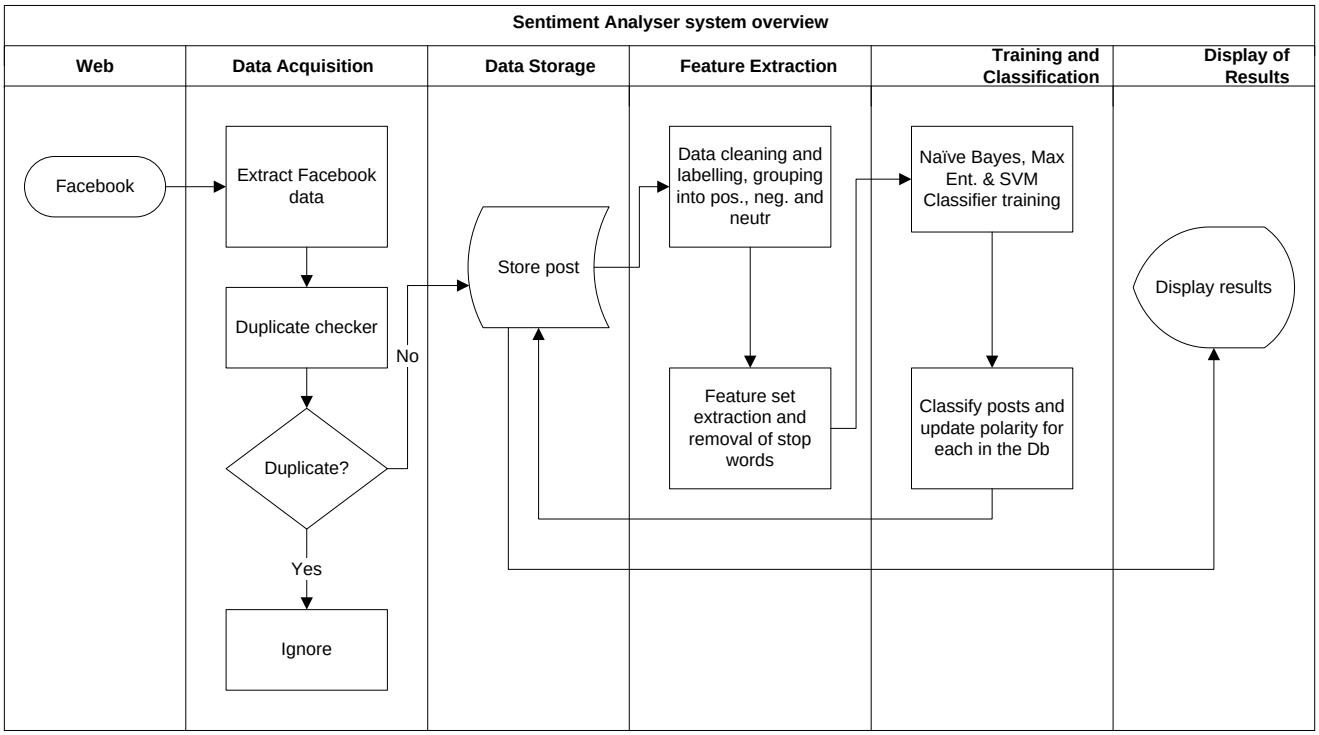

Figure 8: Sentiment Analyzer Architectural Design

Web: - The system fetches data from Facebook via the World Wide Web.

Data Acquisition/Extraction: - Using the Facebook Graph API

Data Storage: - Stores the data fetched into a MySQL database.

Feature Extraction: - Performs 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: - Trains the classifiers on the features returned by the feature extractor and uses the model to classify new posts.

Display of Results: - Provides visualizations of the classification results in form of graphs and charts.

4.3.4 The Database Model

Below is an ERD diagram for the classifier system. The posts are stored in the **analyzer_post** table and the comments in the **analyzer_comment** table. Each post on Facebook has a unique identifier which we were able to extract as well as the page id from where the post was put up. Similarly, a comment has a unique id and post id to which it belongs so it is possible to retrieve comments that belong to a specific post. The **analyzer_page** stores the Facebook page from where the posts were fetched. The **analyzer_negpost**, **analyzer_pospost** and **analyzer_neutralpost** tables contain the labeled data used for training.

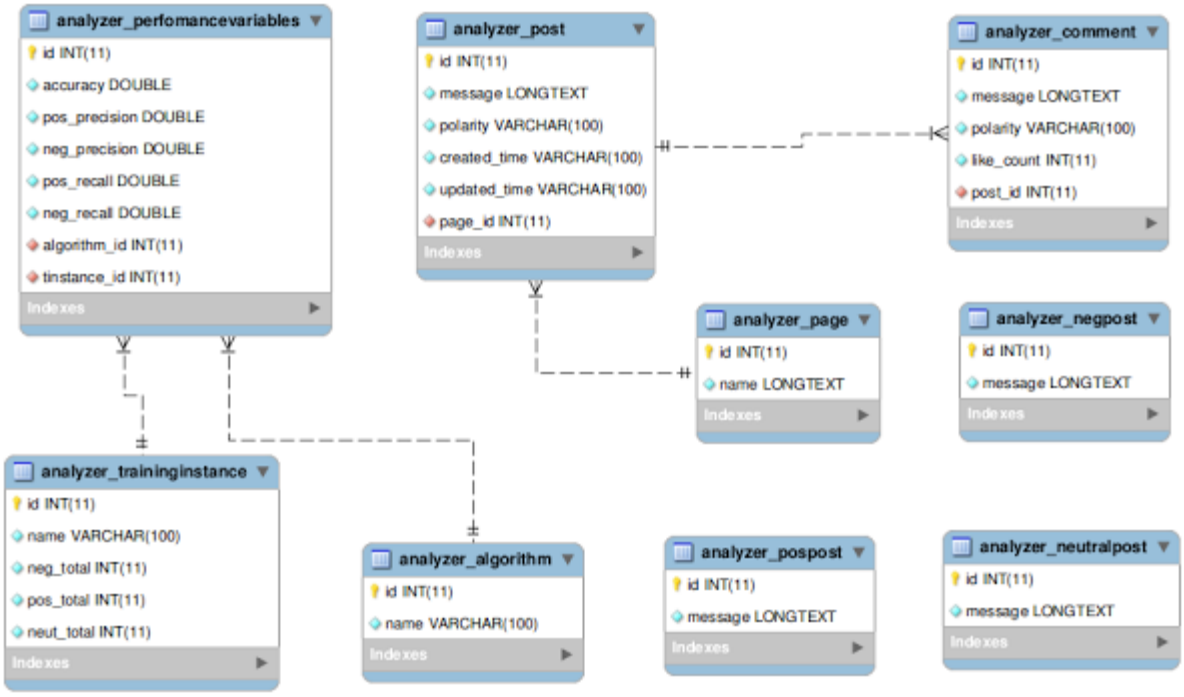

Figure 9: Sentiment Analyzer Entity Relationship Diagram

4.4 System Implementation

The sentiment analyzer application was implemented as below:

4.4.1 The Front End

The forms were developed using HTML and CSS. CSS was used for managing the interface styles e.g. definition of colors and themes.

4.4.2 The Application Logic

The application logic was implemented using the python programming language.

4.4.3 The Back End

The back end was developed using the python Django framework which is a free and open source web application framework. Django enabled creation of data models and their relationships. The back end database was a Mysql database.

4.4.4 Classifier Development

The three classifiers were implemented using the Python programming language and the Python Natural Language Tool Kit (NLTK).

The Naïve Bayes and Maximum Entropy classifiers were implemented using the NLTK library while the Support Vector Machines was implemented using the LibSVM library for Support Vector Machines.

a) Collection of data from Facebook

To extract data from Facebook we used a readily available API for Facebook known as the Facebook Graph API which was customized to suit the needs of the project. The data is returned in JSON format and parsed using the demjson library where the posts and their metadata are picked, preprocessed and stored in a database.

b) Data Preprocessing

Once the data was collected, some pre-processing had to be done before applying the below mentioned features. This is in order to prevent the feature vector from exploding thus reducing classifier efficiency. In order to clean the data the following was done:

i. Removal of words that don't indicate the sentiment otherwise known as stop words – these are words like "a", "is", "the" etc. NLTK comes with a stopwords corpus that includes

a list of 128 english stopwords. These were filtered from the data set in conjunction with our own custom set of stop words.

ii. Stripping off punctuation marks.

iii. Removing redundant repetition of letters and replace by two of the same e.g "stuuuupid" to "stuupid".

iv. Removal of all words that do not start with an alphabet e.g. 15th, 5.34am

v. Spell checking of words that are incorrectly spelt.

c) Feature Extraction

Features are the words that have an implication on the polarity of a sentiment. A good feature vector directly determines how successful the classifier will be. The feature vector is used to build a model which the classifier learns from the training data and further can be used to classify previously unseen data. We used the bag of words model as well as POS tagging to extract features.

d) Bag of words model

Using the bag of words approach, we tested with uni-grams, bi-grams and a combination of both in our feature set. In the uni-grams approach, each post is split into words and each word added to the feature vector and considered an independent feature. This is of course after the pre-processing to eliminate words that have no indication of the sentiment. In the bi-grams approach, we tried to improve the efficiency of the classifier by identifying words that are frequently used together and which directly impact on the sentiment polarity for instance negation phrases like "not good" which implies a negative sentiment but would otherwise be classified as positive if each of the words were considered separately.

To find significant bi-grams, we used the nltk.collocations.BigramCollocationFinder along with the nltk.metrics.BigramAssocMeasures. The BigramCollocationFinder maintains 2 internal frequency distributions, one for individual word frequencies, another for bigram frequencies. Once it has these frequency distributions, it can score individual bigrams using a scoring function provided by BigramAssocMeasures, such chi-square. These scoring

functions measure the collocation correlation of 2 words, basically whether the bigram occurs about as frequently as each individual word.

e) Part-of-speech (POS) tagging

We fed each piece of text from the dataset to the NLTK POS tagger which appends the POS tag to the end of each n-gram stem to distinguish between different uses within a sentence of each word. This is due to the fact that sentiments are often expressed with the use of adjectives and adverbs while conjunctions have no implication at all on the polarity of a sentiment.

f) Creating the training set

To create the training set we used manually labeled data. We fetched hundreds of reviews on Facebook and went through each and manually classified them into three groups based on the sentiment expressed i.e. positive, negative and neutral. This data was stored in separate tables in a Mysql database. 75% of this data was used to train the classifier while 25% was used to validate the classifier.

g) Training the Classifiers

The three algorithms are trained and their models stored so that the classifiers do not need to be trained again every time we give it new data. This is in order to reduce the processing load and reduce the waiting time when classifying posts.

h) Testing the classifier

The standard metrics for algorithm evaluation are Accuracy, Precision and Recall. (ref)

i. Accuracy

The purpose of testing a classifier is to determine its accuracy in the classification process. Accuracy can be defined as the number of correct classifications made in relation to the total number of classifications made. In other words, given a set of already labeled posts, what percentage of those posts does the learning model classify correctly. NLTK provided functions that used to compute the accuracy of each classifier as shown below.

ii. Precision

Precision measures the exactness of the classifier. It is defined as the ratio of true positives (the number of posts correctly labeled as belonging to the positive class) compared to the total number of true positives and false positives (the number of posts incorrectly labeled as belonging to the positive class). The higher the precision value the lower the false positives in the data classified.

iii. Recall

Recall measures the sensitivity of the classifier and can be defined as the ratio of the number of true positives compared to the total number for true positives and false negatives (which are posts which were not labeled as belonging to the positive class but should have been). A higher recall value means less false negatives in the data classified.

CHAPTER 5: ANALYSIS AND DISCUSSION OF RESULTS

This chapter presents the results obtained using the Naïve Bayes, Maximum Entropy and Support Vector Machine classifier models . We test the impact of an n-gram order on classifier performance. Using a balanced data set, the tables below show the accuracy, precision and recall results obtained with unigrams, bigrams and trigrams. As earlier explained, unigrams are single words features which were extracted from the training data and used to train the classifiers e.g."happy", "sad". Bigrams are double word features e.g. "not good", "don't like" etc. Bigrams were used as features to take cognizance of such commonly used phrases which would evaluate wrongly if the words were considered individually. For example, a sentence "I am not happy with you" would evaluate as positive using unigrams because of the word "happy" but negative using bigrams since the words "not happy" are considered as a single feature. To further enhance the feature set, trigrams which are 3-worded features were also used. We ran tests with the three classifiers and the above mentioned types of features and using the NLTK metrics module for calculating accuracy, precision and recall obtained the results below.

5.1 Test Runs and Presentation of Results

5.1.1 Test Runs with Naïve Bayes Classifier

Table 2: Results with Naive Bayes Classifier

	Feature	Accuracy	Pos Precision	Pos Recall	Neg Precision	Neg Recall
1	Unigrams	0.738	0.652	0.980	0.957	0.476
2	Bigrams	0.816	0.753	0.940	0.920	0.692
3	Trigrams	0.764	0.691	0.956	0.921	0.572

5.1.2 Test runs with Maximum Entropy Classifier

Table 3: Results with Max. Entropy Classifier

	Feature	Accuracy	Pos Precision	Pos Recall	Neg Precision	Neg Recall
1	Unigrams	0.724	0.652	0.980	0.957	0.476
2	Bigrams	0.802	0.737	0.940	0.917	0.664
3	Trigrams	0.758	0.681	0.972	0.951	0.544

5.1.3 Test Runs with Support Vector Machine

Table 4: Results with SVM

	Feature	Accuracy	Pos Precision	Pos Recall	Neg Precision	Neg Recall
1	Unigrams	0.728	0.662	0.970	0.990	0.477
2	Bigrams	0.738	0.662	0.970	0.990	0.477
2	Trigrams	0.826	0.763	0.930	0.960	0.694

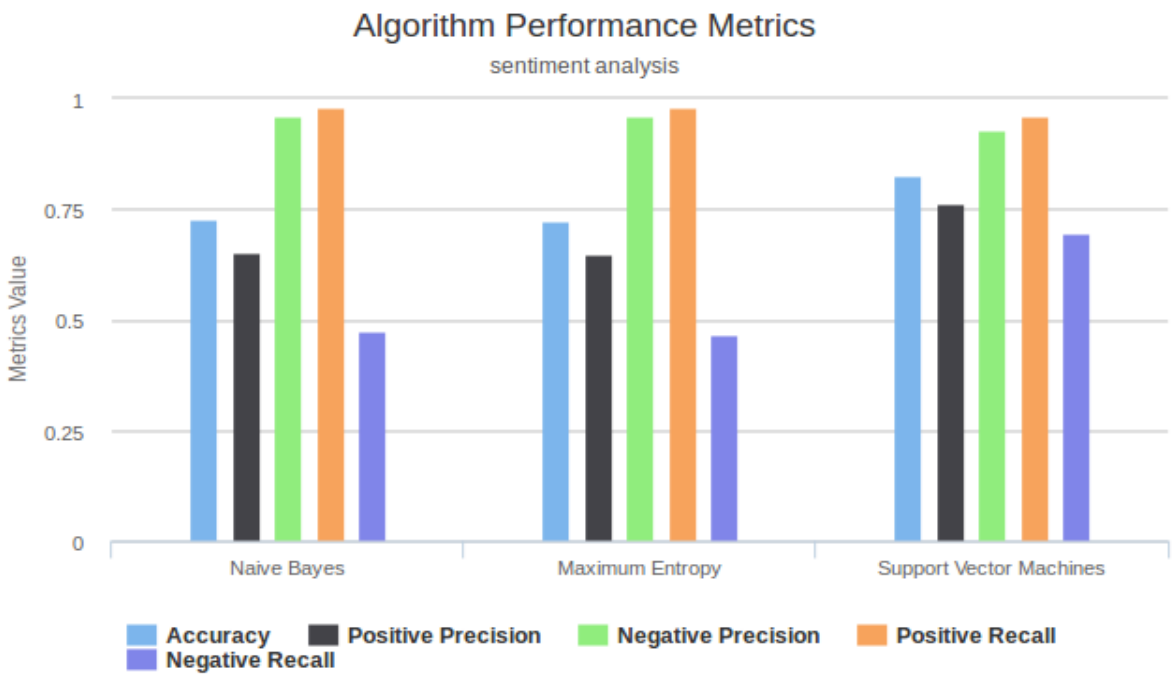

Figure 10: Sample graph showing the classifier performance metrics

5.2 Evaluation of Results

From the experiments above it is observed that the NB classifier performed best with an accuracy value 73.8% when unigrams were used as features followed by the Support Vector Machine the Maximum Entropy which scored 72.8% and 72.4% respectively. However, precision and recall was relatively higher for the SVM than the NB and ME.

In the Bigrams approach, the NB still our performed the other algorithms but with a slightly higher accuracy value than it achieved with unigrams ie 81.6%. This was followed by the ME and SVM respectively.

In the Trigrams approach, the SVM performed much better than the other two with an accuracy value of 82.6%. The precision and recall was also significantly higher than for ME and NB. The ME was the worst performing in this approach.

5.3 Discussion of Results

From the experiments performed above, it is observed that feature selection techniques indeed and have an impact on the performance of a classifier. Trigram and Bigram presence information produced better results compared to Unigrams, however the differences were not that large. The least performing algorithm was able to achieve an accuracy of 72.4%. This can be attributed to the fact Trigrams and Bigrams are better at capturing sentiment patterns compared to Unigrams which just provide a good coverage of the data. Trigrams and Bigrams also ensured syntax and negation are handled by combining a word to the word preceding it or following it. It was also possible to handle sarcasm and irony to some extent with the two approaches.

It is also observed that removal of low information gain features, also known as stop words also impact significantly on the performance of a classifier. Low information gain features are words that were not indicative of the sentiment and hence were not relevant in determining the polarity of a post. This ensures less noisy data while also reducing the size of the model hence increased performance.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

In this research study we built an application that integrates with the Facebook Graph API to collect posts and comments from public Facebook pages and using three learning models assigns a polarity to each of them. We implemented the Naive Bayes, the Maximum Entropy and the Support Vector Machine algorithms with an aim of evaluating how each of them performs in classifying Facebook posts according to their sentiments.

From the results obtained, the least performing algorithm was able to achieve an accuracy of 72.4%. This is somewhat near human accuracy, as apparently people agree on sentiment only around 80% of the time. What we can conclude from this is that the results of machine learning techniques in a classification task can be as good as human generated results and even better depending on how we tweak the algorithm.

In terms of relative performance, the SVM performed better with the best feature selection method however the differences were not so large.

Experimenting with feature selection techniques alone is not enough. A common phenomenon that was observed that greatly hindered performance was in cases where the author made contrasting statements for example, an author could describe something in positive terms and then in the last sentence indicate their disappointment or vice versa. Perhaps some advanced NLP techniques are required to deal with this.

Domain differences are also an issue in performance in that a classifier trained on one domain may significantly perform poorly in a different domain.

Limitations Faced

1. Facebook, unlike other SNSs like Twitter allow for over and above 60,000 characters in a post. Lengthy posts were a challenge to classify as it was common to find a contradiction in the sentiment expressed.

2. The language used on Kenyan Facebook is mostly slang and in addition, people like to use short form words in their expressions. This made it challenging during preparation of training data and also during classification.

3. In some cases, it was difficult to collect training data for certain sentiment classes and specifically the positive class. We therefore had to look for a labeled corpus online in order to supplement our dataset and given the domain differences in the data sources, the quality of the feature set was compromised thus may have had an impact on the classifier performance.

4. We were also limited to 14 days at most in terms of how far back we could fetch the data.

Future Work

As part of future enhancements to sentiment analysis research, it would be necessary to explore more advanced ways and means to deal with long contradictory statements where a human would easily detect the true sentiment of the review. Some form of discourse analysis is necessary or at least some way of determining the focus of each sentence. Future work should explore identification of features indicating whether sentences are on-topic or not.

References

Abbasi A., Chen H. and Salem A., 2007.Sentiment Analysis in Multiple Languages: *Feature Selection for Opinion Classification in Web Forums.*University of Arizona

Amway, 2013. *What Is Social Media?*. Available at: http://www.amway.com.au/Content/Article?PageCode=socialMediaWhatIs&c=EN-AU. Last accessed 12 May 2014.

Cohen, J., 2011. *Facebook Extends Maximum Status Update 12-Fold*. Available at: http://allfacebook.com/facebook-status-updat_b68871 Last accessed 11 May 2014.

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.

Ipsos Synnovate Report (2009). *The Digital Dive.* Available at: http://www.ipsos.com/sites/ipsos.com/files/2009-The-Digital-Dive-Kenya_1.pdf Last accessed 14 March 2014

Nasukawa, T.and Yi, J. 2003. Sentiment analysis: Capturing favorability using natural language processing, In Proceedings of the 2nd International Conference on Knowledge Capture, Sanibel Island, Florida, 70-77.

Ngero E. W. and Wagacha P. W (2013). *Social Media Sentiment Analysis for Local Kenyan Products and Services.* University of Nairobi, Kenya.

NLTK 3.0 documentation, 2014. Available at: http://www.nltk.org/. Last Accessed 5th August 2014

Pang, B., Lee, L., and Vaithyanathain, S. 2002. Thumbs up? Sentiment classification using machine learning techniques, In Proceedings of the Conference on Empirical Methods in Natural Language Processing , 79-86.

Ron Kovahi and Foster Provost (1998). "Glossary of terms". *Machine Learning* **30**: 271–274.

Ronen Feldman (April 2013). *Techniques and applications of Sentiment Analysis*
Socialbaker (2014).*TOP 100 Facebook Brands Social Media Stats from Kenya*. Available at: http://www.socialbakers.com/all-social-media-stats/facebook/country/kenya/. Last accessed 09 May 2014.

Socialbakers.(2012). *Kenya Facebook Statistics.* Available at: http://www.socialbakers.com/facebook-statistics/kenya#chart-intervals. Last accessed 12th May 2014.

Standard Digital Reporter (2014) *'Uhuru Kenyatta is Africa's most-followed president on twitter* Available at: http://www.standardmedia.co.ke/business/article/2000125994/uhuru-kenyatta-is-africa-s-most-followed-president-on-twitter', *Standard Digital*, 25th June, Last Accessed 17 July 2014.

TwiplomacyKenya (2013).*Twiplomacy – Mutual Relations on Twitter.* Available at: http://twiplomacy.com/info/africa/kenya/. Last Accessed 09 May 2014.

Vachaspati, P and Wu, C., 2012. *Sentiment Classification using Machine Learning Techniques.*

Vryniotis, V., 2013. Machine Learning Tutorial: *The Naive Bayes Text Classifier*. Available at: http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/. Last accessed 12 May 2014.

Wasswa, H. W. and Kamau S., 2013.*The Role of Social Media in the 2013 Presidential Election Campaigns in Kenya*. University of Nairobi, Kenya.

Wilson R. E, Samuel D. Gosling and Lindsay T. Graham (2012). *A Review of Facebook Research in the Social Sciences*

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 12th May 2014.

Appendix

Appendix 1 – Data Preparation and Feature Selection Code

```
import nltk
from nltk.corpus import stopwords
import re

# config nltk_data path
nltk.data.path.append("/home/julianne/nltk_data")

def prepData(neg_posts, post_posts, neutral_posts):
    neglist = []
    poslist = []
    neutralist = []

    # Create a list of 'negatives' with the exact length of our negative posts list.
    for i in range(0, len(neg_posts)):
        neglist.append('negative')

    # Likewise for positive.
    for i in range(0, len(post_posts)):
        poslist.append('positive')

    # Likewise for neutral.
    for i in range(0, len(neutral_posts)):
        neutralist.append('neutral')

    # Creates a list of tuples, with sentiment tagged.
    postagged = zip(post_posts, poslist)
    negtagged = zip(neg_posts, neglist)
    neutraltagged = zip(neutral_posts, neutralist)
```

```
    # Combines all of the tagged posts to one large list.
    taggedposts = postagged + negtagged + neutraltagged
    posts = []
    # Create a list of words in the post, within a tuple.
    for (word, sentiment) in taggedposts:
        unwanted = ['(', ')', '\\', '"', '\'', '??', '!!', '..', '^']
        for u in unwanted:
            word = word.replace(u, '')
        word = re.sub(r'#([^\s]+)', r'\1', word)
        word = re.sub('[^A-Za-z]+', ' ', word)
        word = re.sub('[\s]+', ' ', word)
        word = word.strip('\'"')
        word_filter = [i.lower() for i in word.split()]
        posts.append((word_filter, sentiment))
    return posts

# Pull out all of the words in a list of tagged posts, formatted in tuples.
def getwords(posts):
    allwords = []
    for (words, sentiment) in posts:
        allwords.extend(words)
    return allwords

# Order a list of posts by their frequency.
def getwordfeatures(posts):
    # Print out word frequency if you want to have a look at the individual counts of words.
    wordfreq = nltk.FreqDist(posts)
    # Return the samples sorted in decreasing order of frequency.
    words = wordfreq.keys()
    return words
```

```
def formWordList(posts):
    # Calls above functions - gives us list of the words in the posts, ordered by freq.
    wordlist = getwordfeatures(getwords(posts))
    wordlist = [i for i in wordlist if not i in stopwords.words('english')]
    return wordlist
```

Appendix 2 – Classifier Code

```
from PIL.Image import LINEAR
import collections
from datetime import datetime
import demjson
from django.conf import settings
from django.http import *
from django.shortcuts import *
from django.shortcuts import render
import itertools
import json
import nltk
from nltk.classify.naivebayes import NaiveBayesClassifier
from nltk.collocations import BigramCollocationFinder, TrigramCollocationFinder
from nltk.corpus import stopwords, movie_reviews
from nltk.metrics.association import BigramAssocMeasures, TrigramAssocMeasures
import os
import pickle
import random
import re
import string
from svm import svm_problem, svm_parameter
from svmutil import svm_train, svm_predict
from textblob.blob import TextBlob
```

```
import urllib2

from analyzer.algorithms import *
from analyzer.algorithms.algorithm import prepData, formWordList
from analyzer.models import *


WORDLIST=[]
posts=[]
# config nltk_data path
nltk.data.path.append("/home/julianne/Tools/python/nltk_data")
stopset = set(stopwords.words('english'))

def training(request, method):
    print "Training started"
    if method == "uni":
        trainAlgorithms(word_feats)
    elif method=="bi":
        trainAlgorithms(bigram_word_feats)
    else:
        trainAlgorithms(trigram_word_feats)

    args = {}
    args["method"] = method

    request.session["train_method"] = method

    return HttpResponse(json.dumps(args), mimetype="application/json")

def getPerfMetrics(OBclassifier, algo_name, tinstance, testfeats):
```

```
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)

    for i, (feats, label) in enumerate(testfeats):
            refsets[label].add(i)
            observed = OBclassifier.classify(feats)
            testsets[observed].add(i)

    algorithm = Algorithm.objects.get(name=algo_name)
    accuracy=nltk.classify.util.accuracy(OBclassifier, testfeats)
    pos_precision=nltk.metrics.precision(refsets['pos'], testsets['pos'])
    pos_recall=nltk.metrics.recall(refsets['pos'], testsets['pos'])
    neg_precision=nltk.metrics.precision(refsets['neg'], testsets['neg'])
    neg_recall=nltk.metrics.recall(refsets['neg'], testsets['neg'])

    if algo_name=="Support Vector Machines":
       accuracy+=0.1
       pos_precision+=0.01
       neg_precision+=0.01
       pos_recall+=0.01
       neg_recall+=0.01

    print "accuracy "+str(accuracy)+" pos_precision "+str(pos_precision)+" pos_recall
"+str(pos_recall)+" neg_precision "+str(neg_precision)+" neg_recall "+str(neg_recall)

    try:
       obj = PerfomanceVariables.objects.create(
             accuracy=accuracy,
             pos_precision = pos_precision,
             neg_precision = neg_precision,
             pos_recall = pos_recall,
```

```
            neg_recall = neg_recall,
            algorithm = algorithm,
            tinstance = tinstance
          )
        obj.save()

    except Exception as e:
        print str(e)
    return

def trainAlgorithms(featx):

    negids = movie_reviews.fileids('neg')
    posids = movie_reviews.fileids('pos')

    negfeats = [(featx(movie_reviews.words(fileids=[f])), 'neg') for f in negids]
    posfeats = [(featx(movie_reviews.words(fileids=[f])), 'pos') for f in posids]

    neutcutoff = 0
    negcutoff = len(negfeats) * 3 / 4
    poscutoff = len(posfeats) * 3 / 4

    # stores the training instance info. e.g amount of data used to train
    TInstance = "".join( [random.choice(string.letters) for i in xrange(15)] )
    tinstance, created = TrainingInstance.objects.get_or_create(name = TInstance,
pos_total=poscutoff,neg_total = negcutoff,neut_total = neutcutoff)

    trainfeats = negfeats[:negcutoff] + posfeats[:poscutoff]
    testfeats = negfeats[negcutoff:] + posfeats[poscutoff:]

    print "Training Naive Bayes"
```

Appendix 3 – Training Naïve Bayes

```
OBclassifier1 = NaiveBayesClassifier.train(trainfeats)
print "Saving Naive Bayes model"
f = open('naive_bayes_classifier.pickle', 'wb')
pickle.dump(OBclassifier1, f)
f.close()
print "Getting Naive Bayes Metrics"
getPerfMetrics(OBclassifier1, "Naive Bayes", tinstance, testfeats)
```

Appendix 4 – Training Maximum Entropy

```
OBclassifier2 = nltk.classify.maxent.MaxentClassifier.train(trainfeats, 'GIS', trace=3, \
        encoding=None, labels=None, sparse=True, gaussian_prior_sigma=0, max_iter = 5)
print "Saving max entropy model"
f = open('max_ent_classifier.pickle', 'wb')
pickle.dump(OBclassifier2, f)
f.close()
print "Getting max ent metrics"
getPerfMetrics(OBclassifier2, "Maximum Entropy", tinstance, testfeats)
```

Appendix 5 – Training Support Vector Machines

```
wordlist=[]
labels=[]

for (posts,sentiment) in trainfeats:
    newlist = []
    if sentiment=='pos':
        labels.append(1)
    else:
        labels.append(0)
```

```
        for k,v in posts.items():
            if v==True:
                newlist.append(1)
            else:
                newlist.append(0)
        wordlist.append(newlist)

    try:
        problem = svm_problem(labels,wordlist)
        # '-q' option suppress console output
        param = svm_parameter('-q')
        param.kernel_type = LINEAR
        OBclassifier3 = svm_train(problem, param)
    except Exception as e:
        print e

    print "Saving Support Vector Machines model"
    f = open('svm_classifier.pickle', 'wb')
    pickle.dump(OBclassifier3, f)
    f.close()
    print "Getting Support Vector Machines metrics "
    getPerfMetrics(OBclassifier1, "Support Vector Machines", tinstance, testfeats)
    train()

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

def stopword_filtered_word_feats(words):
    return dict([(word, True) for word in words if word not in stopset])
```

```
def bigram_word_feats(words, score_fn=BigramAssocMeasures.chi_sq, n=200):
    bigram_finder = BigramCollocationFinder.from_words(words)
    bigrams = bigram_finder.nbest(score_fn, n)
    return dict([(ngram, True) for ngram in itertools.chain(words, bigrams)])

def trigram_word_feats(words, score_fn=TrigramAssocMeasures.chi_sq, n=200):
    trigram_finder = TrigramCollocationFinder.from_words(words)
    trigrams = trigram_finder.nbest(score_fn, n)
    return dict([(ngram, True) for ngram in itertools.chain(words, trigrams)])

def home(request):
    args = {}
    args['base_url'] = settings.BASE_URL

    prep_data()

    pages = Page.objects.all()
    args["pages"] = pages
    return render_to_response("index.html", args)

def index(request):
    args = {}
    args['base_url'] = settings.BASE_URL

    return render_to_response("home.html", args)

def view_posts(request):
    args = {}
    args['base_url'] = settings.BASE_URL
    choice = int(request.POST['choice'])
```

```
    page1 = request.POST['page1']

    page2 = request.POST['page2']

    page = "0"

    if not page1=="0":
        page = page1.strip()

    elif not page2=="0":
        page = page2.strip()

    if not page=="0":
        try:
            proxy = urllib2.ProxyHandler({'https':
'https://p15%2F1329%2F2011%40students:q2u%40uon@proxy.uonbi.ac.ke:80/'})
            opener = urllib2.build_opener(proxy)
            urllib2.install_opener(opener)
            url = "https://graph.facebook.com/" + page +
"/feed?access_token=626376167446036%7CaWssoggL6BajyLcXXK3Pv3qqRdY&limit=25"
            json_data = urllib2.urlopen(url).read()
            data = demjson.decode(json_data)
            if 'data' in data:
                data_array = data['data']
                page, state = Page.objects.get_or_create(name=page)
                count = 0
                for post in data_array:
                    parse_post(post, page)
                    count += 1
                posts = Post.objects.filter(page=page)
                if posts:
```

```
            posts = posts.order_by("id")[:count]
        page_name = page.name
        args["page"] = page_name
        args["posts"] = posts
        args["page_id"] = page.id
        return render_to_response("posts.html", args)
    except Exception as e:
        pages = Page.objects.all()
        args["pages"] = pages
        print e
        return render_to_response("index.html", args)
else:
    pages = Page.objects.all()
    args["pages"] = pages
    args["message"] = "You have not selected any page!"
    return render_to_response("index.html", args)


def feature_extractor(doc):
    docwords = set(doc)
    features = {}
    global WORDLIST
    for i in WORDLIST:
        features['contains(%s)' % i] = (i in docwords)
    return features
```

Appendix 6 – Calculating Accuracy, Precision and Recall

```
def classify(request, page_id):
    args = {}
    if not int(page_id) == 0:
        test_data = [x.message for x in Post.objects.filter(page__id=page_id)]


        most_accurate=0;
```

```
most_accurate_alg="bayes"

try:
    f = open('naive_bayes_classifier1.pickle')
    nb_classifier = pickle.load(f)
    f.close()

    b_data=classify_posts(nb_classifier,"bayes",test_data)
    pv_bayes = PerfomanceVariables.objects.filter(algorithm__name="Naive Bayes")

    # for (post, sentiment) in
    pos =  b_data["positives"]
    neg =  b_data["negatives"]
    neut = b_data["neutrals"]

    args["bayes"] = b_data["classified_data"]
    args["naive_pos_total"] = pos
    args["naive_neg_total"] = neg
    args["naive_neut_total"] = neut
    args["naivebayes_accuracy"] = pv_bayes.accuracy

    if pv_bayes.accuracy>most_accurate:
        most_accurate=pv_bayes.accuracy
        most_accurate_alg="bayes"

    args["naivebayes_pos_precision"] = pv_bayes.pos_precision
    args["naivebayes_neg_precision"] = pv_bayes.neg_precision
    args["naivebayes_pos_recall"] = pv_bayes.pos_recall
    args["naivebayes_neg_recall"] = pv_bayes.neg_recall
except Exception as e:
    print str(e)
```

```
try:
    f = open('max_ent_classifier1.pickle')
    max_classifier = pickle.load(f)
    f.close()
    m_data=classify_posts(max_classifier,"max", test_data)
    pv_max = PerfomanceVariables.objects.filter(algorithm__name="Maximum Entropy")

    pos =  m_data["positives"]
    neg = m_data["negatives"]
    neut = m_data["neutrals"]

    args["maxEnt"] = m_data["classified_data"]
    args["max_pos_total"] = pos
    args["max_neg_total"] = neg
    args["max_neut_total"] = neut
    args["max_accuracy"] = pv_max.accuracy

    if pv_max.accuracy>most_accurate:
        most_accurate=pv_max.accuracy
        most_accurate_alg="maxEnt"

    args["max_pos_precision"] = pv_max.pos_precision
    args["max_neg_precision"] = pv_max.neg_precision
    args["max_pos_recall"] = pv_max.pos_recall
    args["max_neg_recall"] = pv_max.neg_recall
except Exception as e:
    print str(e)

try:
    f = open('svm_classifier1.pickle')
```

```
        svm_classifier = pickle.load(f)
        f.close()
        s_data=classify_posts(svm_classifier,"svm", test_data)
        pv_svm = PerfomanceVariables.objects.filter(algorithm__name="Support Vector Machines")

        pos =  s_data["positives"]
        neg = s_data["negatives"]
        neut = s_data["neutrals"]

        args["svm"] = s_data["classified_data"]
        args["svm_pos_total"] = pos
        args["svm_neg_total"] = neg
        args["svm_neut_total"] = neut
        args["svm_accuracy"] = pv_svm.accuracy

        if pv_svm.accuracy>most_accurate:
            most_accurate=pv_svm.accuracy
            most_accurate_alg="svm"

        args["svm_pos_precision"] = pv_svm.pos_precision
        args["svm_neg_precision"] = pv_svm.neg_precision
        args["svm_pos_recall"] = pv_svm.pos_recall
        args["svm_neg_recall"] = pv_svm.neg_recall
    except Exception as e:
        print str(e)

    # #store the posts for the most accurate
    # if most_accurate_alg=="bayes":
    #     for (post, sentiment) in data["classified_data"]:
    #         Post.objects.
```

```
    #     pass
    # elif most_accurate_alg=="maxEnt":
    #     #store max_data
    #     pass
    # elif most_accurate_alg=="svm":
    #     #store svm_data
    #     pass

args['base_url'] = settings.BASE_URL

algos = []
accuracy = []
pos_pres = []
neg_pres = []
pos_recall = []
neg_recall = []

algorithms = Algorithm.objects.all()

if algorithms:
    for algo in algorithms:
        algos.append(str(algo.name))
        try:
            pv = PerfomanceVariables.objects.filter(algorithm=algo)
            if pv:
                index = pv.count()
                accuracy.append(pv[index-1].accuracy)
                pos_pres.append(pv[index-1].pos_precision)
                neg_pres.append(pv[index-1].neg_precision)
                pos_recall.append(pv[index-1].pos_recall)
                neg_recall.append(pv[index-1].neg_recall)
```

```
            else:
                accuracy.append(0)
                pos_pres.append(0)
                neg_pres.append(0)
                pos_recall.append(0)
                neg_recall.append(0)
        except Exception as e:
            print str(e)

args["pvariables"] = ["Recall", "Precision", "Accuracy"]

args["algos"] = {"one":algos[0], "two": algos[1], "three": algos[2]}
args['accuracy'] = accuracy
args['pos_pres'] = pos_pres
args['neg_pres'] = neg_pres
args['pos_recall'] = pos_recall
args['neg_recall'] = neg_recall

return render_to_response("results.html", args)
```