UNIVERSITY OF NAIROBI

COLLEGE OF BIOLOGICAL & PHYSICAL SCIENCES

SCHOOL OF COMPUTING AND INFORMATICS

MSC PROJECT

AN APPROACH FOR USING TWITTER TO PERFORM SENTIMENT ANALYSIS IN KENYA

BY - ERIC GITAU
P58/9104/2006

SUPERVISOR – EVANS MIRITI

FEBRUARY, 2011

SUBMITTED IN PARTIAL FUILFILMENT OF THE REQUIREMENTS OF THE MASTERS OF SCIENCE IN COMPUTER SCIENCE

DECLARATION

I declare that the work contained in this report, is my own work except where explicitly indicated in the text and has not been previously submitted in another university for any award.

Name: Eric Gitau **Reg. No:** P58/9104/2006

Signed: [illegible] **Date:** 08/07/2011

This project has been submitted as part fulfillment of the requirement for the Masters of Science in Computer Science, University of Nairobi, with my approval as the University Supervisor.

Name: EVANS MIRITI **Signed:** [illegible]

Date: 08/07/2011

ACKNOWLEDGEMENTS

I would like to begin by thanking the Almighty God for granting me his grace, favor, strength and wisdom during the time spent in developing this piece of work.
I would like to thank my supervisor, Mr. Evans Miriti for his suggestions, advice and guidance during the project period. I wish to extend my gratitude to Dr. Wanjiku Ng'ang'a, for her excellent ideas and suggestions which have greatly impacted the shape and form of this project. I wish to also thank Mr. Robert Oboko of the project examination panel for his technical input into the work that is presented in this document as well as his encouraging words.
Lastly my gratitude goes to my family, friends and colleagues; my parents for their encouragement, support and patience as I labored on this project, my colleagues, for their support and understanding nature, and in particular, Mr. Kirwa Kemei for his technical assistance during the initial stages of writing code for this project, Stella Mitine, for her support and encouraging words, and my employer for KRA for providing an environment where I could take time off to work on this project.

To you all I say Thank You.

ABSTRACT

The interest in sentiment analysis as a research area has become increasingly popular with the development of new social interaction technologies. Twitter, being one of these new technologies, presents a unique environment where one can track sentiments expressed about various topics. This report therefore considers the problem of attempting to classify sentiments expressed on twitter about certain products, services or personalities as being positive, negative or neutral. The approach adopted to solve this problem is through the use of machine learning methods. In particular, the Naïve Bayes model is chosen to build the classifier. This being a learning problem, training data and testing data is required. Two methods of collecting training data are considered and their impact on the performance of the classifier is discussed. The first method is distant supervision, where emoticons are used as labels to identify and collect training data that contains sentiment information. The other method is manual supervision where a human trainer manually identifies and labels training data with that contains the necessary sentiment information. It is discovered that using distant supervision to collect training data results in poorer performance, than using manual supervision techniques, even where the training set collected using distant supervision is larger than the training set from the manual supervision techniques. Using emoticons as labels to identify 5000 tweets as training data, the classifier performed with an accuracy of 70.3% compared to use of 500 hand labeled tweets as training data which resulted in 76.3% accuracy. A third method for collecting training data using manual supervision methods is also suggested and its performance is also discussed. This method which uses hand labeled keywords grouped according to word characteristics yields a performance of 80.3%. This report concludes by giving recommendations of ideal models to start with when attempting to develop a twitter based sentiment classifier. A software tool, developed using the learning model to classify live streams of data from twitter into positive, negative or neutral classes and provide a summary of results, is also demonstrated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

NLTK	Natural Language Tool Kit
NB	Naïve Bayesian
SVM	Support Vector Machines
ME	Maximum Entropy
ND	Noisy Data
SMS	Short Messaging Service.
IRC	Internet Relay Chat
POS	Part of Speech
API	Application Programming Interface
PHP	PHP Hypertext Processor
JSON	JavaScript Object Notation
URL	Uniform Resource Locator
ELTD	Emoticon Labeled Training Data
HLTD	Hand Labeled training Data
KWTD	Key Word Training Data
NLP	Natural Language Processing
NLU	Natural Language Understanding

I. INTRODUCTION

A. Background

Twitter is a social network that allows registered user to communicate to other members through updates to their personal profile pages with small status messages of less than 140 characters. This can be done through a web interface or through SMS on mobile phone devices. Since its inception, the nature of twitter has undergone tremendous transformation. The end result has been a micro-blog where users express certain emotions, ideas, feelings and personal sentiments on a wide array of topics ranging from sports and movies, to brands, products and services. The ease of use of twitter, and its fun nature makes it popular among the young and those technologically savvy. It is no surprise therefore, that a lot of big companies have twitter accounts through which they communicate and engage their customers in an attempt to better understand customers' needs while improving the mode of communication between them as service providers and the consumers of their products and services.

A lot of times, marketing departments are curious about what their customers think about their products or services. Traditional methods of gathering this information would be through market research in the form of a survey which is usually a time consuming and costly activity. Twitter on the other hand is one of the social spaces where such information can be easily collected. Users are constantly using twitter to express their opinions on all manner of subjects. Even better is the fact that these opinions are expressed immediately they are experienced. The use of mobile devices makes it possible to interact with twitter while you undertake other activities. This makes it possible to share feelings and opinions immediately you experience something. For example a lady while out shopping for shoes enters a shop and sees a pair she likes. She immediately logs in to her twitter account and updates her status message about where she has seen the shoes and expresses her opinion on the shoes. This information is valuable to the shoe designer because he wants to know what people think about his shoes, and the owner of the shoe store who is interested in how many people mention his shop.

Given a certain subject of topic, the challenge then is to be able to collect opinions expressed by different people in various forms and media, and provide a summary of their general opinion. Sentiment analysis is an area of research that attempts to address this challenge. It tries to determine the general thoughts of a group of people in relation to a certain subject matter. The main task performed during analysis of sentiments is the classification of user opinions into certain categories, where the category with the most number of opinions is considered to be the preferred sentiment. This is an area of research that is closely linked to data mining, and it uses computer science disciplines such as machine learning, especially in the domain of natural language processing, to come up with algorithms and models that can be used to deduce opinions from people's comments about specific topics.

With the large amount of data found on twitter spurred by its growth and popularity, and the growing need for information about products and services, a platform can be developed that combines these two areas, thereby providing a way for any interested party to get an idea on what the twitter community feels about a certain product, service or personality.

B. Definition of Concepts

Tweet: - A message update to a profile page of a registered Twitter user.

Emoticon: - An emoticon sometimes also known as smiley is a combination of standard punctuation symbols and letters of the alphabet that is used to signify an emotion in a piece of text. Emoticons have their origin in IRC chat systems and newsgroups where face to face communication was not possible over the internet. Users therefore started denoting their emotions through this semi-graphic text and symbolic combinations. An example of these emoticons include :-) for happiness, and :-(for sadness.

C. Problem Statement:

The use of twitter as a social network in Kenya is currently growing and with this growth, a lot of potentially useful information is being passed through the network. This information can provide some value to marketers or researchers on the general perception of their products or services. However the nature of tweets from Kenyan users is such that both English and Swahili text is usually combined in the message, and this combination is usually unstructured and informal. Currently, the few sentiment analysers that exist have only been trained in English. This therefore means that attempting to get an opinion on a product or service name in Swahili will produce erroneous results. Similarly attempting to search for a Kenyan product in English might also produce an erroneous result. This is because Swahili data was not used to train the classifiers. Therefore any tweets with a mixture of English and Swahili would be ignored, and this would then skew the results in favor of one side of the sentiment. There is therefore need to develop a sentiment analyser that is trained in both English and Swahili therefore correctly classifying tweets that contain both English and Swahili, as well as allowing searches of products or services that are named in Swahili.

It is also not very clear on the best approach to take when building such a learning model. Classification of tweets for sentiment analysis is a new research area; hence the various models that exist for other classification problems, may or may not produce desired results when applied to this problem. There is therefore a need to compare the suitability of existing learning models for purposes of using tweets for sentiment analysis.

D. Objective

The purpose of this project therefore is to:

- Analyze machine learning models used in other classification problems and discuss their suitability for the problem of classifying tweets for sentiment analysis before suggesting a good starting point.
- Using a suitable model, develop a classifier, that uses data collected from twitter in both English and Swahili as training data, and then extract features that will allow it to classify sentiments in tweets as positive, negative or neutral.
- Based on the above classifier, develop a web based application that does sentiment analysis on a search criterion using real time data from twitter on people, products and services that are found in the Kenyan market. The search items are to be entered in either English or Swahili.

E. Significance of the Project:

The proposed project is significant because it will involve work that would lead to:-

- An understanding of the challenges faced in sentiment analysis and ways to handle these challenges.
- The discussion of a possible scientific starting point, for the problem of sentiment analysis where twitter is the source of training data.
- The creation of a learning model that will be integrated with software, which will be able to handle query searches in Swahili and English and also contain training data in both languages.
- The development of a demo showcasing how a platform for getting feedback on products or services in the Kenyan market using data from twitter can be created.
- Laying of the foundation for further study that can incorporate other social networks therefore increasing the sample space which in turn would improve the quality of data used for the sentiment analysis. E.g. a researcher interested in performing sentiment analysis on Facebook can use the results of this research to get an idea of the most ideal feature extraction methods to use for Facebook. The training data collected during this research can also be reused thereby reducing the amount of time it would take to develop the Facebook sentiment analysis tool.

II. LITERATURE REVIEW

A. Introduction to Natural Language Processing

NLP is an area of research in computer science falling under the wider subject of machine learning. Machine learning in itself can be defined as a the process of creating computer systems that are able to use human like cognitive skills to learn new patterns and behaviors that they can then use to make future decisions and produce certain desired results. Machine learning involves the process of "learning" where a human "teacher" uses training data to familiarize the machine with expected output behaviors, after which the machine, when presented with new data to processes using the learned knowledge, is able to derive new results. NLP is defined as an area of research that attempts to use computational techniques to give computer systems the ability to read pieces of text understand the meaning implied in the text and use the information gathered to make decisions and perform some form of future processing. For successful NLP research, the researchers aim to understand how natural languages are constructed and using that knowledge provide the same abilities for computer systems. Being a subset of machine learning, NLP uses tools and techniques derived from various other disciplines such as mathematics, statistics and artificial intelligence. This makes it possible to apply NLP technologies to various applications and research areas. Some of these areas include information retrieval for online search engines, and sentiment analysis. In order to solve a problem using NLP, an important element to be considered is Natural Language Understanding. Using NLP requires understanding of natural language so that the model is able to deduce meaning from the text presented before it can use the knowledge to perform any meaningful processing. The process of Natural Language Understanding (NLU) can be understood as a mathematical process of mapping text into classes or entities that show the relationship between the pieces of text.

1. Statistical and Corpus Based Methods for NLP

As earlier mentioned, NLP borrows from various academic areas with mathematics and statistics specifically playing a vital role in implementation of learning models. In the area of information retrieval for example, common mathematical techniques used include use of probability distributions and markov models. These mathematical models have given rise to various techniques used as part of processes necessary for NLP such as word sense disambiguation, parsing, etc. Statistical and corpus based methods are mainly used for natural language understanding. These methods became popular between the 80's and 90's as collections of text became more common. A corpus is simply a large collection of text usually structured in a specific format. Using corpus based methods allowed for use of statistical methods. Since the corpus has structure it becomes possible to model the variables necessary for creating relationships used in the learning process. The corpus based approach therefore reduced the complexity present in trying to create learning models.

B. <u>Sentiment Analysis</u>

The field of sentiment analysis is a wide research area covering a number of different approaches, methodologies and algorithms. The last ten years have been instrumental in the growth of the research field mainly due to the increasing interest in machine learning methods among researchers, coupled with the massive growth of internet usage all over the world. Social media has become one of the biggest areas of internet business providing large amounts of information that can be leveraged for purposes of sentiment analysis. Similarly emerging technologies are fuelling innovative techniques of information collection, classification and dissemination allowing this research area to grow in terms of collection of training data and presentation of sentiment results.

One of the earliest mentions of the term "sentiment analysis" is found in the paper by Das et al (2001) where they attempt to derive market sentiment from stock message boards. Subsequently the term became more common, with most researches opting to use it over the term "opinion mining" when discussing a more technical approach towards the process or extracting sentiments from text documents. The use of machine learning techniques in sentiment analysis marked the beginning of a more scientific approach towards this problem. Pang et al (2002) carried out various tests using different machine learning algorithms and compared their results to human classification methods. They used a movies review corpus to train and test their classifiers after which they were able to classify movie reviews as either positive or negative. There results indicated that the statistical methods were able to outperform the human classification methods in terms of accuracy. The best learning model from the three they studied was the SVM model with an accuracy of 82.9% when using unigrams as features while checking for feature presence. The SVM model was also the best performer when both unigrams and bigrams were combined giving score of 82.7%.The NB classifier had the best result with accuracy of 78.7% when using unigrams as training features and classifying based on feature frequency. The ME classifier also took home some glory during the tests by being the best model when using adjectives, and bigrams as features. It registered scores of 77.7% and 77.4% respectively. The work done by these researchers created a great place to start when deciding to use machine learning models for sentiment analysis.

C. <u>Sentiment Analysis on the Web</u>

As the use of machine learning techniques became more understood the shift of sentiment analysis now moved to the application of these models on data found on the internet. Zhang et al (2007) discussed the process of retrieving opinion from blogs by using machine learning based classifiers (SVMs) to perform sentence by sentence classification. Similarly Yang et al (2006) described a method used to retrieve opinion from blog postings by employing a fusion approach. They first used information retrieval methods to retrieve blogs containing certain topics of interest, after which they used opinion extracting methods to weight the blogs depending on the degree of opinion expressed. The growth of internet use and social media also presented new possibilities for sentiment analysis. The use of emoticons in online chat tools, blog postings and social networks provides an identifier that can be used to capture pieces of text containing sentiment information. Read (2005) used emoticons to identify training data that was used to train and develop a sentiment classifier. The classifier performed with an accuracy of up to 70% when predicting the sentiment of articles extracted from the emoticon based dataset. However, the performance was unsatisfactory when used to classify external pieces of text, such as news articles. This was largely attributed to fact that the size of the articles being classified was considerably large hence the vocabulary created during the training of the classifier was insufficient for accurate classification.

D. Language Processing for Sentiment Analysis

Another interesting area of research that is always closely associated within sentiment analysis is linguistics. Problems of text classification using machine learning models require an understanding of how language is constructed and what to look for in order to get the best results during classification. The understanding of languages can therefore provide new ideas on how best to use language constructs, and language structure to extract features for sentiment analysis. For example Benamara et al (2007) attempt to show that use of adjective and adverbs in training data is better than using adjectives alone for sentiment analysis. This they do this buy considering various adverb adjective combinations while focusing on adverbs of degree. Such discussions can spur creative ideas on how best to integrate other existing disciplines, technologies and methodologies for purposes of sentiment analysis.

E. Twitter as a Corpus for sentiment Analysis

Coming closer to our problem of sentiment classification using twitter as a corpus, Go et al (2009) described an approach of using distant supervision to collect training data used to train a classifier for purposes of classifying tweets as belonging to either a positive or negative class. Their approach used emoticons as a noisy label used only to identify tweets containing sentiment information, after which the emoticon was stripped from the tweet and the rest used as part of the training set. They then trained 3 machine learning based classifiers (NB, ME and SVMs) and compared the accuracy of results when used to classify hand labeled test data. The best performance was realized by the ME classifier with an accuracy of 83% when they combined both unigrams and bigrams as training features. The NB classifier produced the highest accuracy when bigrams were used as the only training feature with an accuracy of 81.6%, while the SVM classifier had the highest performance of 82.2% and 81.9% when using unigrams and unigrams combined with part of speech tags as training features respectively. Pak et al (2009) used a similar approach where they used emoticons as noisy labels to identify tweets that contained sentiment information. Using the NB classifier, they then classified tweets as either positive or negative based on either bigrams or POS tags as word features. The use of bigrams as the feature extraction method produced the best accuracy.

F. Conclusion

Looking at the two cases of using twitter as a corpus for sentiment analysis, a number of opportunities for further research arise. The first area is the introduction of a neutral class in the classification process. This class will improve classification of live tweets where the classifier is unable to make a proper classification. The assumption made here is that some of the errors made during classification occur as a result of the classifier not knowing where to classify neutral text. For example tweets containing "pass along" information to news stories or links to other websites do not necessarily have any sentiments in them. However in a training set that contains English stopwords, the classifier may classify such a tweet as either positive or negative depending on the probabilities of observing those stopwords in the training sets. This means that proper handling of the neutral class is important in ensuring the accuracy of overall results presented by the classifier.

Another important area of further research is extending sentiment analysis using twitter to the use of a multilingual corpus. This will allow the extraction of sentiments from tweets that are posted in more than one language. The research described in this report attempts to incorporate methods that will create this feature by using a training set containing both English and Swahili words. In both languages, formal and informal language constructs are considered. This is to cater for the largely informal nature of conversations on twitter and also the constraints of using less than 140 characters to communicate. The general twitter population is also generally made up of younger users hence the use of slang is most preferred.

Finally it is important to ask ourselves whether using distant supervision affects the accuracy of the classifier. By simply collecting tweets that contain emoticons, and then assuming that there are sufficient features in those tweets to be used to classify sentiment in tweets without emoticons, one may end up with a learning model that consumes lots of training data, yet fails to deliver in terms of accuracy when put to the literal test of classifying live data. This research hopes to compare the performance of a classifier trained with data collected through distant supervision using emoticons vis-à-vis a classifier trained with data collected through normal supervision by using hand labeled data.

III. METHODOLOGY

A. Introduction

The problem of classifying tweets as either a positive or negative can be solved through the use of machine learning algorithms as discussed in the previous chapter. In order to implement a learning algorithm, the following components are required.

- The training data
- The testing data

1. Training Data

The training data is defined as the data that the learning algorithm uses to extract training features, which are subsequently used in the classification of new data. This training data is commonly comprised of text that is previously labeled as belonging to either one of the classification classes. The use of the training data is to come up with the training set. The training set is usually made up of class labels, which identify to what sentiment class the set itself belongs, as well as the training features themselves. Training features could include unigrams, bigrams, and part of speech tags among others. The training features are usually grouped according to the sentiment class where they were extracted from. By doing this, the learning algorithm is able to know which training features will be used to identify a specific sentiment class.

2. Testing Data

The purpose of testing data is to provide where data where testing set will be extracted as to be used by the learning algorithm to test the accuracy of the classification process. The testing data is comprised of text that is previously labeled as belonging to either of the sentiment classes. From the testing data, a testing set is extracted in a similar fashion to the extraction of the training set. Thus the testing set will have also have features such as bigrams, unigrams etc. The learning algorithm uses features extracted from the training set previously discussed above, to classify input from the testing set, and then compares its classification results, with the initially labeled classification. Where the two classes are similar, then an accurate classification has been realized, otherwise the learning algorithm has made an error in the classification process.

B. Naïve Bayes Classifier for Text Classification

In the previous chapter, it was seen that the use of the Naïve Bayes model yields considerably good results when used to classify pieces of text especially where bigrams are the training features. Coupled with its relative ease of implementation whence compared with other classification models, the naïve bayes model presents a good starting point for twitter based sentiment analysis using machine learning methods.

A naïve bayes classifier is a machine learning model used to classify a document as belonging to one of a set of classes by computing probabilities using the bayes rule. It is used in a subset of machine learning methods called supervised learning where a human "teacher" trains the classifier on how to make classifications. Given an input document X, the naïve bayes classifier will classify X to belong to a class Y from a set of classes (Y1, Y2, Y3,...., YK) based on previous training data.

From the Bayes Rule:

$$p(x,y) = p(y \mid x).p(x) = (px \mid y).p(y)$$

$$p(y \mid x) = \frac{p(x \mid y).p(y)}{p(x)}$$

The probability of determining the class Y given X can be computed from the formula above. To do this it is necessary to compute $p(x \mid y)$. For this purpose, naïve bayes makes the assumption that the probability $p(x_i \mid y)$ is independent of all other probabilities of (x_i) found in the piece of text. This is known as the conditional independence assumption and is represented below.

$$p(x \mid y) = \prod_{k=1}^{n} p(x_k \mid y) = p(x_1 \mid y) \times p(x_2 \mid y) \times \ldots \times p(x_n \mid y)$$

For classification to happen, $p(y_i \mid x)$ is computed for all values of $(x_i \mid y_i)$. Two main models of naïve bayes exists. The Multivariate Bernoulli model defines $p(x_i \mid y)$ as the number of documents of class (y) where (x_i) appears. On the other hand the multinomial model defines $p(x_i \mid y)$ as the number of times (x_i) appears in the all documents labeled as belonging to class (y)

The decision to classify x into class (y_i) will be determined by the larger of the two scores of $p(y_i \mid x)$.This can be represented as

$$C_{NB} = \arg\max_y p(y \mid x) = \arg\max_y \left(\prod_{k=1}^{n} p(x_k \mid y) \right) \times (p(y))$$

In order to use a naïve bayes classifier, a training set is required. This is usually in the form of a vocabulary made up of words found in the documents that make up the training set. These words are assigned a label to denote to which class they belong. From the training set, the classifier extracts the training features. These features include unigrams, bigrams etc. When the classifier is presented with a new tweet to classify, it would have to extract features from the tweet and calculates the probability of observing in the training set, all the extracted features, with respect to both positive and negative classes. The classifier would then classify the tweet based on which class returns the highest probability.

For the purpose of classifying tweets as either positive or negative, a definition of the naïve bayes formula is given with the following variables.

$$C_{NB} = \arg\max_y p(y \mid x) = \arg\max_y \left(\prod_{k=1}^{n} p(x_k \mid y) \right) \times (p(y))$$

Where

$y = (y_1 = positive, y_2 = negative)$ - This set defines the positive or negative class.

$x = (x_1, x_2, \ldots, x_n)$ - These are the words found in each tweet.

$p(x_i \mid y)$ - This defines the probability of classifying word (x_i) found in tweet (x) as a member of class (y).

$p(y)$ - The probability of observing the class y.

1. Collection of Training Data

Training data as was earlier mentioned is an important requirement in the development of a learning algorithm. For the purpose of classifying tweets as either positive or negative, a training data in the form of tweets is required. A number of approaches are available for acquiring these tweets.

a) Method 1: Use of Emoticons as Noisy Labels.

With the rise in popularity of social media outlets such as blogs, social networks and media sharing portals, users continue to find new ways to express feelings and opinions about the media they consume. One of the most popular ways to do this is through the use of emoticons. Twitter is one social space that makes extensive use of emoticons. With the limitation of 140 characters per status updates, people will generally wish to convey their feelings in the shortest from possible. Emoticons therefore feature extensively on twitter as a way of expressing opinion.

The use of emoticons as noisy labels for identifying tweets with sentiment information was discussed by Go et al (2009). They defined a procedure where they simply searched for tweets with either positive or negative emoticons, collected the results and created a training data containing refined tweets from the search results. This appears to be a good place to start based on the results they achieved using this method. For the collection of tweets for training data, the made is that assumption that any tweet containing positive emoticons contains a positive sentiment within the text. Similarly, in any tweet where any negative emoticon is found, then that tweet contains sentiment information of a negative nature. This therefore means that for the collection of training data, what is required is to search twitter for positive and negative emoticons and store all tweets that the search returns.

Data Clean Up

One the tweets are collected, some data clean up is to be performed on the data before any features can be extracted from the set. Go et al (2009) found that leaving the emoticons in the training set would affect the accuracy of the results of the learning model. For this reason, all emoticons need to be removed from the training set during the data clean up stage. Similarly, http hyperlinks all twitter based symbols (such as @ for usernames, # for hash tagged twitter topics) need to be removed from the training set as well because they are generally common in most tweets.

Feature Extraction

Once the data has been cleaned up, the training features can now be extracted from the training data. A number of feature extraction methods can be considered.

- Use of unigrams: - using this method, each word present in the tweet is considered as an independent feature and is therefore collected as a training feature for the sentiment class where the tweet occurs. The end result of using unigrams to extract training features is a training set that is similar to a "bag of words" where all the words that are found in the tweets collected in the training data make up the training set.
- Use of bigrams: - this feature extraction method identifies each bigram present in the tweet as an independent feature. The bigram is stored as part of the training set in the sentiment class where the tweet occurs.

b) Method 2: Use of labeled data.

The second approach for collecting training data is the traditional use of manually labeled data. For our case, there would be need to hand label tweets as belonging to either the positive of negative class. To identify such tweets, a list of 5 common keywords was identified. The idea is to query twitter using each of these keywords and observe the resulting tweets. Where a tweet contains sentiment information, the tweet is placed in its respective class as identified by the sentiment information.

Data clean up.

Once hand labeled tweets are collected, data cleaning should also be performed on the set in a similar fashion to method 1 above, with only a slight modification. While in method 1 we did propose to do away with emoticons since they were noisy labels, in this method, we plan to do away with the query search terms present in the tweets. This is because the presence of a query term in many tweets that contain a certain sentiment class would result in that query term being identified as a feature during the training process.

Feature Extraction

Feature extraction for labeled data follows similar steps as extraction of features for emoticon labeled data. Both unigrams and bigrams are identified and stored in their respective training set classes.

2. **Collection of Testing Data**

Another important element of a learning algorithm as described earlier is the testing data. The use of the testing data will make it possible to extract a testing set which allow one to determine whether the classifier is making the correct classification. In order for one to create a testing set of tweets, an ideal approach is to use query terms to collect a number of tweets. These tweets would then be analyzed and those with sentiment information in them are hand labeled and placed in their respective sentiment classes. Once a substantial number of tweets are realized in both classes, the testing data is complete. No data cleaning is necessary for the testing data. This is because the test data needs to closely resemble the live data that the learning model will be classifying once the process of developing the classifier is complete. However, the process of extracting features from the testing data will be similar to the process described for extracting features from the training data. Both unigrams and bigrams will be considered in this case.

C. <u>Collection of Data from Twitter:</u>

For the purpose of classifying tweets as either positive or negative, a training set and testing is required to train and test the classifier respectively. During this chapter we have extensively talked about the collection of tweets for this very purpose. The most reasonable approach for this activity is the use of the twitter search API. The twitter API provides an interface that allows one to query twitter for a search term and return tweets containing the search keyword. These tweets are rate limited to a maximum of 1500 tweets per call, for a maximum period of 4 days since the time of the search. Further to this, the API allows inclusion of a geocode variable into the query, so as to limit the search results to a certain locale. A twitter query can therefore be constructed with variables such as the latitude and longitude values of Nairobi as the geocode, and the relevant search keywords as the query variable, returning a result set containing all tweets containing the keyword mentioned in Nairobi within 4 days of the search.

IV. IMPLEMENTATION OF THE EXPERIMENT

A. Introduction

In the previous chapter, we have looked at the methodology for creating a machine learning model that can be used to classify tweets as either positive or negative. One of the necessary ingredients of a machine learning model is the training set. We discussed two approaches of collecting training data for purposes of creating a training set. In this chapter we shall be looking at the actual processes used to actualize the methodology previously discussed, while paying attention to any similarity or difference found in the results obtained from using two different training data collection methods.

B. Implementation of the learning model

The model chosen in this case was the Naïve bayes model with the development environment being the NTLK in python. Since the aim was to look at performance and accuracy, it was not necessary to implement a personal Naïve bayes model. The decision was made to use the naïve bayes classifier implemented by NLTK. This was done by importing the *nltk.classify* package. This package would then allow us to call methods such as *classifier.train(testfeats)*-which trains the classifier using the variable testfeats which represents the training features. More details about the python environment will be discussed later on in this chapter.

C. Collection of Data from Twitter

Collection of data from twitter was made possible by the use of a search tool. The search tool was developed in PHP through the use of a reusable PHP-Twitter integration library. (Please see the appendix for further reference). This library contained class constructor methods for the various objects that the twitter API has made public. This made it possible to easily instantiate these classes with the local variables required without having to define all the classes again. The information exchange between the search API and the search tool developed was handled through the use of the JSON format. Queries were created using URL's and passed to the API as http get methods while the results from the API to the search tool were transmitted through the JSON format. The JSON format was chosen since the PHP-Twitter wrapper class chose to use JSON format as the transport protocol of choice. The figures below illustrate the process of retrieving tweets through the search tool.

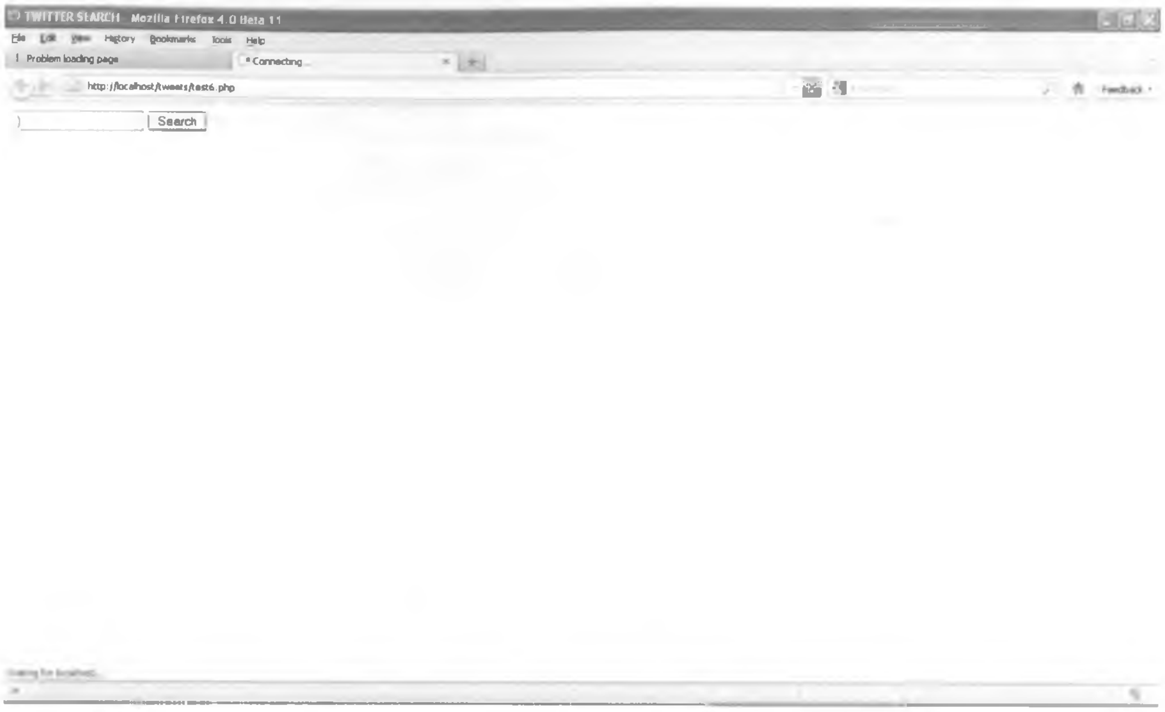
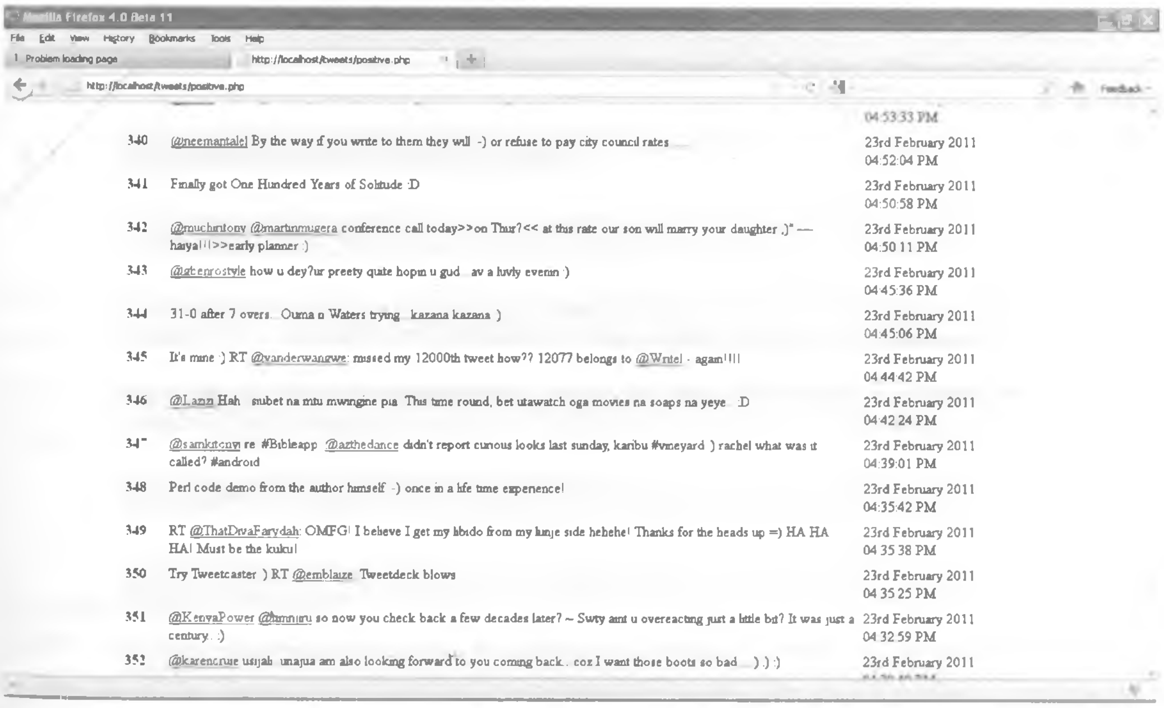
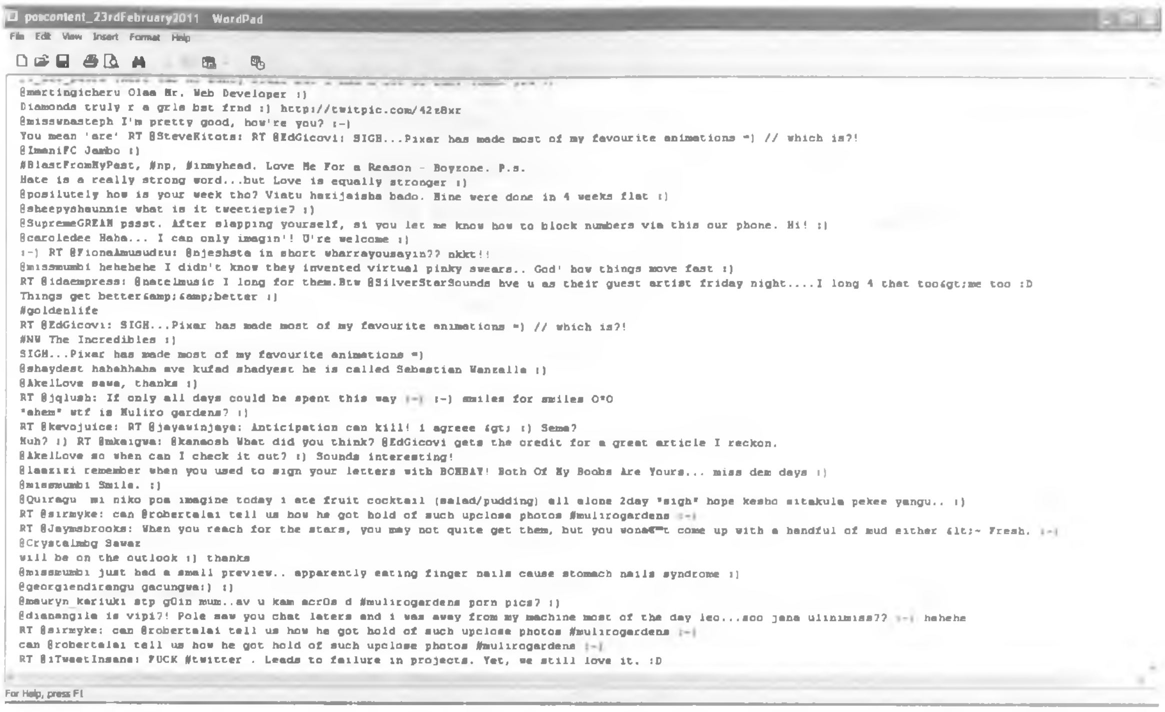

TWITTER SEARCH - Mozilla Firefox 4.0 Beta 11
File Edit View History Bookmarks Tools Help
Problem loading page
Connecting...
http://localhost/tweets/test6.php
Feedback
)
Search

Figure 1 :- An emoticon used as the input to a twitter search query


Mozilla Firefox 4.0 Beta 11
File Edit View History Bookmarks Tools Help
Problem loading page
http://localhost/tweets/positive.php
Feedback

		04:53:33 PM
340	@neemantalel By the way if you write to them they will -) or refuse to pay city council rates	23rd February 2011 04:52:04 PM
341	Finally got One Hundred Years of Solitude :D	23rd February 2011 04:50:58 PM
342	@muchiritony @martinmugera conference call today>>on Thur?<< at this rate our son will marry your daughter ;)" — haiya!!!>>early planner :)	23rd February 2011 04:50:11 PM
343	@gbenrostyle how u dey?ur preety quite hopin u gud av a luvly evenin :)	23rd February 2011 04:45:36 PM
344	31-0 after 7 overs. Ouma n Waters trying kazana kazana :)	23rd February 2011 04:45:06 PM
345	It's mine :) RT @vanderwangwe: missed my 12000th tweet how?? 12077 belongs to @Wntel - again!!!!	23rd February 2011 04:44:42 PM
346	@Lann Hah subet na mtu mwingine pia This time round, bet utawatch oga movies na soaps na yeye :D	23rd February 2011 04:42:24 PM
347	@samkitonyi re #Bibleapp @azthedance didn't report curious looks last sunday, karibu #vineyard :) rachel what was it called? #android	23rd February 2011 04:39:01 PM
348	Perl code demo from the author himself :-) once in a life time experience!	23rd February 2011 04:35:42 PM
349	RT @ThatDivaFarydah: OMFG! I believe I get my libido from my luhya side hehehe! Thanks for the heads up =) HA HA HA! Must be the kuku!	23rd February 2011 04:35:38 PM
350	Try Tweetcaster :) RT @emblaize Tweetdeck blows	23rd February 2011 04:35:25 PM
351	@KenyaPower @kimniru so now you check back a few decades later? ~ Surly aint u overeacting just a little bit? It was just a century. :)	23rd February 2011 04:32:59 PM
352	@karencruse usijali unajua am also looking forward to you coming back.. coz I want those boots so bad ;) ;) :)	23rd February 2011

Figure 2 :- Results from the emoticon query



```
@martingicheru Olaa Mr. Web Developer :)
Diamonds truly r a grls bst frnd :) http://twitpic.com/42z8xr
@misswnasteph I'm pretty good, how're you? :-)
You mean 'are' RT @SteveKitota: RT @EdGicovi: SIGH...Pixar has made most of my favourite animations =) // which is?!
@ImaniFC Jambo :)
#BlastFromMyPast, #np, #inmyhead. Love Me For a Reason - Boyzone. P.s.
Hate is a really strong word...but Love is equally stronger :)
@posilutely how is your week tho? Viatu hazijaisha bado. Mine were done in 4 weeks flat :)
@sheepyshaunnie what is it tweetiepie? :)
@SupremeGREAN pssst. After slapping yourself, si you let me know how to block numbers via this our phone. Hi! :)
@caroledee Haha... I can only imagin'! U're welcome :)
:-) RT @FionaAmusudzu: @njeshsta in short wharrayousayin?? nkkt!!
@missmumbi hehehehe I didn't know they invented virtual pinky swears.. God' how things move fast :)
RT @idaempress: @natelmusic I long for them.Btw @SilverStarSounds hve u as their guest artist friday night....I long 4 that too&gt;me too :D
Things get better&amp;&amp;better :)
#goldenlife
RT @EdGicovi: SIGH...Pixar has made most of my favourite animations =) // which is?!
#NW The Incredibles :)
SIGH...Pixar has made most of my favourite animations =)
@shaydest hahahhaha ave kufad shadyest he is called Sebastian Wanzalle :)
@AkelLove sawa, thanks :)
RT @jqlush: If only all days could be spent this way :-) :-) smiles for smiles O*O
*ahem* wtf is Muliro gardens? :)
RT @kevojuice: RT @jayawinjaye: Anticipation can kill! i agreee &gt; :) Sema?
Muh? :) RT @mkaigwa: @kanaosh What did you think? @EdGicovi gets the credit for a great article I reckon.
@AkelLove so when can I check it out? :) Sounds interesting!
@laazizi remember when you used to sign your letters with BOMBAY! Both Of My Boobs Are Yours... miss dem days :)
@missmumbi Smile. :)
@Quiragu  mi niko poa imagine today i ate fruit cocktail (salad/pudding) all alone 2day *sigh* hope kesho sitakula pekee yangu.. :)
RT @sirmyke: can @robertalai tell us how he got hold of such upclose photos #mulirogardens :-)
RT @Jaymsbrooks: When you reach for the stars, you may not quite get them, but you wonâ€™t come up with a handful of mud either &lt;~ Fresh. :-)
@Crystalmbg Sawaz
will be on the outlook :) thanks
@missmumbi just had a small preview.. apparently eating finger nails cause stomach nails syndrome :)
@georgiendirangu gacungwa:) :)
@mauryn_kariuki stp gOin mum..av u kam acrOs d #mulirogardens porn pics? :)
@dianangila is vipi?! Pole saw you chat laters and i was away from my machine most of the day leo...soo jana ulinimiss?? :-) hehehe
RT @sirmyke: can @robertalai tell us how he got hold of such upclose photos #mulirogardens :-)
can @robertalai tell us how he got hold of such upclose photos #mulirogardens :-)
RT @iTweetInsane: FUCK #twitter . Leads to failure in projects. Yet, we still love it. :D
```

poscontent_23rdFebruary2011 WordPad
File Edit View Insert Format Help
For Help, press F1

Figure 3 :- Results stored in a text file

The tweets were automatically stored in a text file for further processing.

In order to perform a search, the query passed to the API was constructed in the following format:

GEThttp://search.twitter.com/search.json?geocode=-1.274359%2C36.813106%2C15.0mi&q=%3A%29&rpp=100&page=1 HTTP/1.1

Here we can see the query term ":)" is represented in its URL encoded format as "*q=%3A%29*", while the geocode variable is "*-1.274359%2C36.813106%2C15.0mi.*" This defines the coordinates of Nairobi and returns tweets within 15miles of this coordinates. "*rpp=100*" describes a rate of 100 tweets per page, while "*page=1*" shows that the results being retrieved currently are the first page of a maximum of 15 pages.

D. <u>The Python Environment</u>

The entire process of creating the training and testing sets, training the classifier, testing the classifier, and creating the actual classifier was done in python using the NLTK. The NLTK contains a collection of functions and methods that can be used to perform various natural language processing tasks in an easy way. It was not the intention of this project to develop our own implementation of a learning model. The NLTK provides an ideal set of tools that allow rapid prototyping of solutions and was thus deemed the best development platform. The main methods to consider in this case are:

- The CategorizedPlaintextCorpusReader method: - this method instantiates a categorized corpus reader. The corpus reader is simply a function that takes text documents classified in more than one category and creates lists containing all words appearing in a certain category. The corpus reader is also able to identify class labels, based on the names assigned to the locations of the text e.g. names of folders where text files are located can serve as class labels where each folder contains text for each class. The corpus reader also requires file extensions so that it's able to identify which files form part of the data it is required to load.
- The NaiveBayesClassifier method: - this method made it possible to create a naïve bayes classifier that could be trained, tested and used to classify external data. This was made possible by simply passing training, testing and live data to the classifier function as variables.
- The Bigrams Method: - this method made it possible to extract bigrams from pieces of text by simply passing a list of words as a variable to the bigram function.
- The Metrics Method: - the metrics method provided a means of computing metrics that were necessary for testing the quality of the classification process. This method used the classifier and testing features as its input variables.

E. Creating the Training Set

As already previously mentioned before, two methods were considered for the collection of the training data.

1. Method 1: Using Emoticons as Noisy Labels.

The rationale behind using emoticons as noisy labels was discussed in the previous chapter. The first step in this process was to make a search on twitter with the query being either the positive or negative emoticon. All collected tweets were gathered and stored in two separate text files, one for each sentiment category. Once all tweets were stored in their respective text files, the two files were then placed in a location where the categorized corpus loader would then read the tweets into the python environment. This was done by passing the text files as variables into the corpus loader function within python. The corpus loader then returned each individual tweet as an independent instance of the sentiment category. Class labels were also initialized at this stage since the corpus loader was able to identify two separate folders (one named positive and the other negative) in the location where the tweets were stored.

Feature extraction

With the text files now loaded, the next step was to extract training features. In some cases, before training features could be extracted, it was necessary for data cleaning to be performed on the data. Data cleaning was done so as to compare performance of the classifier with and without the noisy data. The process involved removing stopwords, symbols, hyperlinks and noisy labels such as the emoticons themselves. The development environment of python allowed these two activities of clean up and extraction of training features to be combined into one function

a) Extraction of Unigrams as features

For the extraction of unigram features, the following function was used.

```
def good_word_feats(words):
                    return dict([(word, True) for word in words if not(word.lower().strip(",").strip("!") in ["r",
"u", "ur", "urs", "we", "was", "2", "at", "that", "it", "for", "is", "i", "on", "of", "am", "i'm", ":D", ":P", "D",
"P", ":", ".", ";", "!", "&", ",", "...", ":)", ":(", ":-)", ":-(", ")", "(", "=", "=)", "^_^", "-", "$lt;3", ":/", ":|",
"www","&quot;", "?",",", "--", "&gt;", "&gt;&gt;", ";)",":'(", ";/", ";\\", "'", "=d", "in", "the", ":d", "!!", "+",
"!!!", ":p", "&gt", "&gt:", ":o", ";o", "&lt;3", "a", "rt"]) and (not word.lower() in
stopwords.words('english')) and (not word.lower().isdigit()) and (not word.lower().startswith("#")) and (not
word.lower().startswith("@")) and (not word.startswith("RT")) and (not word.lower().startswith("http"))]
```

For every tweet passed into this function, the result is simply a dictionary data structure that returns all words found in the tweet as long as the word does not begin with "#" or begin with "@" or begin with "http" or the word is not ":)" etc. The python dictionary is a container that stores tuples in the form of a of a key, value mapping. Breaking it down further, the following example shows how a single tweet would have unigram features extracted using this function.

The input tweet is: "SIGH...Pixar has made most of my favourite animations =)"

Before passing this tweet into the unigram function, the first thing to be done would be to break it down into a list of words. This was done by a python function that took a piece of text and split it into words wherever white spaces occurred. This means that in the example above, the output would be a list with 9 elements in it, denoted as

['SIGH...Pixar', 'has', 'made', 'most', 'of', 'my', 'favourite', 'animations', '=)'].

This list, it is then passed as input into the unigram extractor function. Assuming the tweet was identified as one with positive sentiment information, the function would the return the following

[{'SIGH...Pixar',True} {'has',True}, {'made',True} {'most',True} {'of',True} {'my',True} {'favourite',True} {'animations',True} Pos].

The above list of tuples contains one less word since the emoticon at the end of the tweet has been removed. It is also evident that each word present in the tweet is also mapped to a Boolean value of true. This is done so as to make it possible for the python classifier function to correctly identify which features are members of a certain sentiment class. In this case these 8 features are all members of the positive class as denoted by the "Pos" keyword at the end.

This process was repeated until all features were extracted from all tweets present in both the positive and negative classes.

b) Extraction of Bigrams as features

For extraction of bigram features, a similar function was developed that returned each bigram in the tweet, paired with a boolean value of True to represent the membership of that bigram in a certain sentiment class. The following function was used to extract bigram features:

```
def bigram_word_feats(words):

        mybigrams = bigrams(words)
        return dict([(bigram, True) for bigram in mybigrams])
```

A tweet such as "Finally watched TRON LOVED IT :)", is labeled as positive, and would yield the following features:

[{"Finally watched", True} {"watched TRON", True} {"TRON LOVED", True} {"LOVED IT", True} {IT :), True} pos]

In similar fashion to the extraction of unigrams, the bigram features are placed in a list with the keyword "pos" identifying them as members of the positive set of training features. Again this iterative process was performed until all bigram features were extracted from all tweets in both the positive and negative classes.

2. Method 2: Use of labeled data

The other method earlier identified for the collection of training data was through the use of hand labeled data. As discussed in the first part of this chapter, the collection was done through the twitter search tool. For labeled data, a number of query topics were identified. For each of those topics, a search was performed and from the resulting tweets, an analysis was done to determine those with sentiment information. Each tweet with sentiment information was then placed in a separate text file corresponding to its appropriate sentiment class. At the end of this process, there were two text files corresponding to both the positive and negative classes. The last step before having the text files initialized by the python environment was stripping the actual query terms from the tweets. This was done using a simple find and replace procedure where the query keywords were replaced by white spaces. The process of feature extraction in this method was similar to that of method one. Both unigrams and bigram features were extracted using functions identical to those identified in method 1 above. The table below shows some of the common query terms used to search for tweets with sentiment information.

Table 1:- List of common queries terms

Query	No of Positive tweets Collected	No of Negative Tweets Collected
Safaricom	22	42
Zuku	19	26

KPLC	N/A	152
Churchill	25	N/A
safcom	34	40
Homeboyz	10	N/A

F. Training the Classifier

In order to train the classifier, it was necessary to create a new instance of a Naïve Bayes classifier and pass the training features as variables into this function. This was easily achieved through the following code snippet.

```
trainfeats = negfeats + posfeats
classifier = NaiveBayesClassifier.train(trainfeats)
```

The first line combines both the negative and positive training features into a list know as trainfeats. These training features are used in the second line as variables to the train method of the NaiveBayesClassifier object that is now being instantiated into our classifier.

G. Creating the testing set

The procedure used to obtain a test set involved the same activities as that of developing a training set. The first step was to use the search tool to gather tweets. A list of query keywords was identified and then used to query twitter for tweets. The data gathered was analyzed and tweets with sentiment information were placed in the respective text files for those classes. The two resulting text files were then used as input to the corpus loader function in python. Once the corpus loader had created an instance of each tweet, the tweets were passed to the feature extractor functions where unigram and bigram features were extracted from the tweets in similar fashion to the methods described in the previous section.

H. Testing the Classifier

1. Accuracy

The main purpose of testing the classifier was to determine the accuracy of the classification process. Accuracy can be defined as the number of correct classifications made in relation to the total number of all classifications made. That is given a set of already labeled tweets, what percentage of those tweets does the learning model classify correctly. Python provided functions that were able to easily measure the accuracy of the classifier. The example below shows how the accuracy was returned using the functions

```
testfeats = negtestfeats + postestfeats
print 'accuracy:', nltk.classify.util.accuracy(classifier, testfeats)
```

The first line of this snippet simply combines the positive and negative test features into a list of all test features. The next step uses the utilities package of the python NLTK module to calculate the accuracy of the classifier, given the test feats (note that classifier variable present here refers to the same classifier we earlier initialized when training the classifier). The results are then printed to the screen. While the accuracy tends to be the most common metric for measuring the performance of the classifier, other metrics are also necessary in getting a clearer understanding of how the learning model is working. These are the precision and recall metrics.

2. **Precision**

The precision metric measures the preciseness of the classifier. This means that it measures how exact the classifier is in its classification. It is defined as the ratio of the number of true positives compared to the total number of true positives and false positives. The higher the precision value the lower the false positives in the data classified

3. **Recall**

The recall metric measures the sensitivity of the classification and can be defines as the ratio of the number of true positives compared to the total number of true positive and false negatives. A higher recall value means less false negative in the result set of classified data.

4. **The Error Set**

Another important component used during the testing of the classifier is the error set. The error set is simply a list of all tweets that were incorrectly classified during the testing process. The set will usually show what classification the learning model gave in comparison to the label that had been manually set. With a large training and testing set, the error set is important because when critically analyzed, it is possible to observe similarities between pieces of text that are incorrectly classified. For example close investigation may reveal the presence of a certain stop word that is common in a lot of the incorrectly classifier tweets. In this project, the error set was used to identify presence of features that the classifier had not come across and hence unable to correctly classify. Such tweets were then automatically added into the training data for those features to be extracted.

5. **Manual Testing**

The final element in the process of testing the classifier is the use of manual observation. Closely related to the error set discussed above, the manual observation was crucial because of the ability of the human observer to easily identify where a classification error had been made. The person observing the tweets was then able to isolate the causes of the improper classification and take corrective measures. A good example of this is the tweet

"Watching tpf season 4…loving it so far". At face value, this tweet looks like one that would automatically be classified as positive. However, the sentiment classifier may not make a correct classification of this tweet. Why u ask?? Remember we had earlier said that during extraction of features, the tweet is broken down into constituent words by splitting the text where white spaces occur. In this tweet, "4...loving" would be treated as a single word feature. The learning model would not be able to make any classification based on this feature since it does not exist in the training set. Therefore a process of removing such tokens was necessary so that "loving" can be included as an independent feature. For such cases, using manual observation proved to be the most suitable and effective method of carrying out tests.

I. Handling of the Neutral Class

Sentiment analysis cannot be complete without handling the neutral class. When dealing with tweets, the idea of the neutral class cannot be over emphasized enough because of the nature of information exchanged on twitter. Pass along information such as links to news articles, mentions by media houses, questions or general comments etc need to be correctly classified so as to ensure the subjects of these discussions do not reap sentiments where they have not sown, so to speak. In order to handle the neutral class, the approach taken involved the extraction of features from the tweet and then searching for the presence of any of these features in both the positive and negative training feature sets. Where none of the tweet's features were found in the training features, it was assumed that the tweet contained no sentiment information hence it was classified as neutral.

V. EVALUATION AND DISCUSSION OF RESULTS

In the previous chapter, the technical implementations of the suggestions presented in the methodology chapter were discussed. This chapter looks at the results obtained using these implementations, and evaluates the results.

A. Evaluating the Performance of the trained classifier

As you may recall, on of the most important metrics for computing the accuracy of a learning model is the accuracy. In our methodology, two approaches for obtaining training data had been identified. One was collecting tweets using emoticons as noisy labels for sentiment classes, while the other was collecting tweets manually hand labeled with their sentiment classes. For comparison of performance, two naïve bayes classifiers were created. Classifier one (C1) used training data identified to contain sentiment by the use of emoticons, also referred to as emoticon labeled training data (ELTD). Classifier two (C2) used training data manually labeled as belonging to specific sentiment classes, what is also referred to as hand labeled training data (HLTD). The ELTD set contained 5000 tweets, while the HLTD set contained 500 tweets. Testing was performed on both classifiers using a set of HLTD. This was necessary since HLTD closely resembles the live data that learning model would be used to classify in future. The training set was made up of 300 tweets. In each of the classifiers, both unigrams and bigrams were considered as feature extraction methods. We further note that for each feature extraction method, two instances were considered. The first instance contained noisy data (examples include emoticons, symbols, and English stopwords) within the training set while in the second instance, all noisy data (ND) was removed. The combination of both unigrams and bigrams was also considered as a feature extraction method. Once the training features were extracted, the two classifiers were trained and tested and a summary of the results are presented below.

Table 2 :- Summary Performance Results

			FEATURE EXTRACTION METHOD					
Training Data type	**Amount of Training Data**	**Amount of Test Data**	**Unigrams**		**Bigrams**		**Unigrams + Bigrams**	
			With ND	Without ND	With ND	Without ND	With ND	Without ND
ELTD	5000	300	69.3%	70.3%	57%	50%	68.6%	70%
HLTD	500	300	75%	76.3	59.3%	55%	74.6	75.3

B. Discussion

1. Emoticons as Noisy Labels

The use of ELTD resulted in poorer performance than the use of the HLTD even when the amount of training data from the former was ten times more than the latter. In our case ELTD had 5000 tweets while HLTD had a mere 500 tweets, yet the performance was around 5% more using all three feature extraction methods. These results confirm what is manually observable when using emoticons to collect tweets. A lot of the tweets that contain emoticons need not have any sentiment information in them. Most of these tweets are usually greetings and informal chatter or jokes. For example, when attempting to collect training data using emoticons from twitter during the month of December 2010, most of the tweets with an emoticon were accompanied by phrases such as "Merry Christmas", "happy holidays" etc. While the volume of the tweets collected was large only about 5% to 10% of those tweets contained any features that could be extracted for training the classifier. Using emoticons to identify tweets with sentiment information would require the intervention of a human being to filter out all the noisy tweets before the data set can be of any meaningful use. It is more advisable to go with hand labeling tweets, as this would improve performance drastically. Given that with only 500 tweets their was a growth in performance by 5%, getting the HLTD set to 5000 tweets would defiantly improve the performance of the classifier by leaps and bounds

2. Feature Extraction Methods to be used.

A rather interesting thing to note in the results presented is the behavior of the feature extraction methods. Bigrams, traditionally known to be the best method to use with a naïve bayes classifier were outperformed by unigrams. The explanation for this could be the nature of tweets. With tweets having less than 140 characters, and most people using extensive slang on twitter, it may come as no surprise that unigrams would be a much better choice when selecting a feature extraction method for use in classifying tweets. The presence or absence of noisy data in the tweet doesn't seem to make a very substantial difference in the accuracy, expect when using bigrams. This makes sense because adding or removing a stop word before or after certain words, completely changes the sentiment implied by the word.

C. Further Analysis

From the results presented in table 3, it was evident that there was still a lot to be done in trying to improve the quality of the classifier. With this in mind, a last attempt was made at trying to further push the accuracy levels up. A new approach was introduced. This new approach was the use of keywords training data (KWTD). The rational behind this was that looking at already hand labeled tweets, it was possible to identify which keywords implied sentiment in the tweet. Similarly human beings know intuitively which words are commonly used to imply certain sentiments. Therefore the task of collecting training data for this approach came down to using personal intuition to create a list of positive and negative keywords. Since tweets are not so large, it was possible to also manually scan tweets and where there was sentiment information the keywords responsible for that sentiment were extracted and added to the list. A training set of 1165 keywords was developed. Using this training set, training features were extracted and a third classifier was trained using this data. Classifier C3 was tested using the 300 hand labeled tweets used to test C1 and C2 previously. The results of the test performed on C3 are presented below.

Table 3 :- Performance results for the third classifier

	FEATURE EXTRACTION METHOD					
Training data type	Unigrams		Bigrams		Unigrams + Bigrams	
	With ND	Without ND	With ND	Without ND	With ND	Without ND
Use of KWTD	77.6%	77.3%	69%	57.3%	80.3%	77.3%

From the results it is evident that the use of keywords as the training data further improves the quality of the classifier giving us the highest accuracy values so far. The use of keywords for the creation of the training data also played a big part in making the classifier more ideal for Kenyan use. By allowing the human supervisor to manually add keywords intuitively, more and more Swahili words and local street slang became part of the training corpus. This would greatly improve the accuracy of the classification made where the tweets contain Swahili or slang.

D. Implementing the Sentiment Analysis Tool

The previous chapters of this project have been discussing implementations of various components. From data collection from twitter, to training and testing the classifier, pretty much most of the technical work covered during this project has been discussed. However it is not all complete without talking about the sentiment analysis tool. This is web based software that demonstrates twitter based sentiment analysis using live data. It is made up of a search interface that allows a user to enter a search topic and receive results summarizing the overall sentiment of that topic based on discussions on twitter. The system queries twitter for the search topic within Nairobi, using the API methods discussed at the beginning of chapter 4. Each tweet that is received is sent to the sentiment classifier where it is classified as being positive, negative or neutral. Once all tweets are classified, the system computes the totals and provides summarized results. These results are presented as pie charts showing percentage distribution of tweets among the three classes, bar graphs showing distribution of tweets by number, and a timeline that shows overall sentiment behavior of the three classes over a period of 4 days. An example is shown below. The interface design and presentation format is inspired by the work of Go et al (2009).

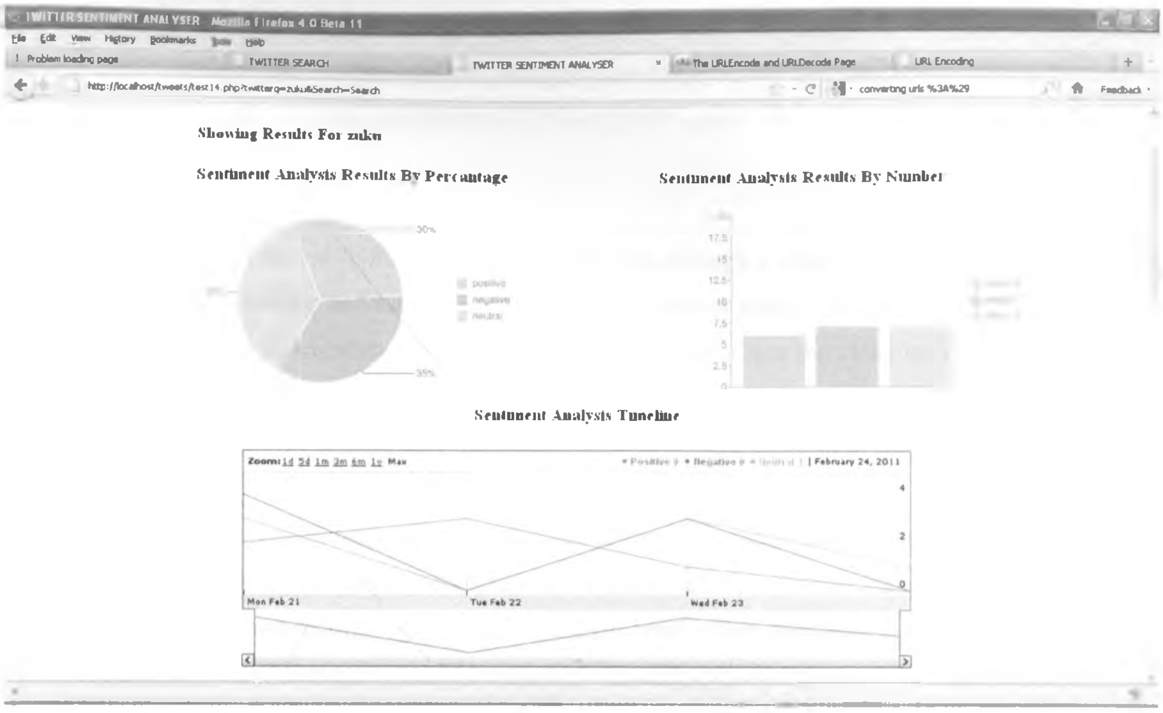
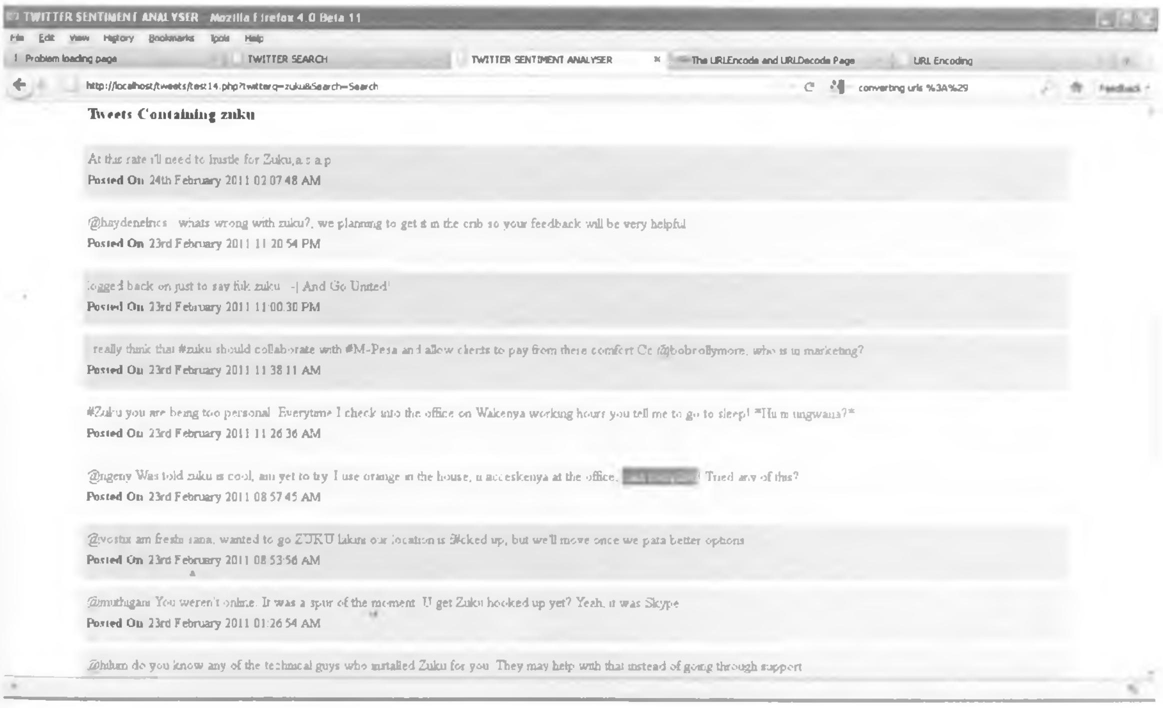

TWITTER SENTIMENT ANALYSER Mozilla Firefox 4.0 Beta 11
http://localhost/tweets/test14.php?twitterq=zuku&Search=Search
Showing Results For zuku
Sentiment Analysis Results By Percentage
Sentiment Analysis Results By Number
30%
35%
positive
negative
neutral
Sentiment Analysis Timeline
Zoom: 1d 5d 1m 3m 6m 1y Max
February 24, 2011
Mon Feb 21
Tue Feb 22
Wed Feb 23

Figure 4 : - the results interface of the sentiment analyser

A set of tweets whose sentiment has been identified is also displayed below the summarized results



TWITTER SENTIMENT ANALYSER Mozilla Firefox 4.0 Beta 11

http://localhost/tweets/test14.php?twitterq=zuku&Search=Search

Tweets Containing zuku

At this rate i'll need to hustle for Zuku,a s a p
Posted On 24th February 2011 02 07 48 AM

@haydenelnes whats wrong with zuku?, we planning to get it in the crib so your feedback will be very helpful
Posted On 23rd February 2011 11 20 54 PM

logged back on just to say fuk zuku -| And Go United!
Posted On 23rd February 2011 11:00.30 PM

really think that #zuku should collaborate with #M-Pesa and allow clients to pay from there comfort Cc @bobrollymore, who is in marketing?
Posted On 23rd February 2011 11 38 11 AM

#Zuku you are being too personal. Everytime I check into the office on Wakenya working hours you tell me to go to sleep! *Hu m ungwana?*
Posted On 23rd February 2011 11 26 36 AM

@ngeny Was told zuku is cool, am yet to try I use orange in the house, n acceskenya at the office, [illegible] Tried any of this?
Posted On 23rd February 2011 08 57 45 AM

@vostux am fresh sana, wanted to go ZUKU lakini our location is f#cked up, but we'll move once we pata better options
Posted On 23rd February 2011 08 53:56 AM

@muthugani You weren't online. It was a spur of the moment. U get Zuku hooked up yet? Yeah, it was Skype
Posted On 23rd February 2011 01:26 54 AM

@hilum do you know any of the technical guys who installed Zuku for you They may help with that instead of going through support

Figure 5 :- some tweets with their sentiment classified

The sentiment analysis tool is implemented in PHP just like the search tool. A number of libraries are used. The pie chart, bar graph and the annotated timeline are components of the Google Visualization API. This is a java script API that allows for the easy creation of charts for purposes of dynamic web based presentation of information. Despite the API being in java script, this tool re uses a PHP wrapper class that was written to allow the easy integration of the API with PHP pages.

Data Cleaning

Before each live tweet is sent to the classifier, it must goes through a clean up process. The words in the tweet are stripped of leading and trailing symbols and punctuation marks. Where these punctuation marks occur in the middle of words, they are replaced by white spaces. Once the clean up is done, the tweet is ready to be passed to the classifier for classification.

Classification

The process of classification begins with extraction of features using function similar to those used to extract training and testing features. The live features returned are then used as input to the classifier which makes a classification and returns a class label. This class label is then submitted back to the PHP front-end for where it is stored for subsequent use in computing overall sentiment scores.

VI. CONCLUSION

In the previous chapter, various results obtained after creating different classifiers each using training data collected using different methods have been discussed. Software that allows users to enter query topics and retrieve sentiment scores concerning those topics, based on data from twitter, has also been demonstrated. With the popularity of twitter increasing, there will always be need to continue further in the field of sentiment analysis on twitter and other forms of social media. This research has presented an approach that can be used to create a baseline, from which further research may be carried out. The use of hand labeled training data should be considered as the lowest starting point for any kind of research where the solution requires a supervised learning model.

A. Achievements

In the first chapter, a number of objectives were identified. The following is a recap of these objectives and to what extent they were met during this project

1. Comparison of Machine learning models:

While it would have been appropriate to practically compare various machine learning methods for purposes of creating a classifier to be used to classify sentiment in tweets, the approach adopted in this case was using previous literature, where the results and conclusions from these, formed a starting point for the development of the classifier. This therefore means that comparison of machine learning methodologies was done theoretically and not practically as earlier envisioned.

2. Development of the Classifier:

This objective was achieved by the implementation of the naïve bayes classifier created using NLTK. The classifier was able use training and testing data collected from twitter via their search API. The tests carried, out whose results are presented in this document, show that the classifier performed to some acceptable level of accuracy making it possible to state that this objective was achieved.

3. Development of the Web Based Application.

The final part of this piece of work was the web based sentiment analysis tool that was created as a demonstration of the classifier. The software was able to take search terms in both Swahili and English, and return real time classified tweets from twitter where the search term occurred. The tweets returned were classifier as positive, negative or neutral.

B. Limitations of the study

The following are some of the limitations of the study undertaken.

1. Collection of training data

While the twitter API made it considerably easy to collect data, there were some challenges that were encountered. The amount of training data that was used is considerably small. Time constraints made it challenging to collect a large amount of training data which is necessary for a concrete machine learning experiment. The tests were localized to Kenya and therefore the amount of 'meaningful' tweets generated by the users of twitter in Kenya is relatively small in comparison to other places where this research has been carried out. Rate limiting enforced by twitter on their search API also meant that only a maximum number of 1500 tweets was available for a 5 day period.

In some cases, it was difficult to collect training data for certain sentiment classes. Specifically, the positive class proved mostly affected. The negative class seemed to contain on average 30% more tweets than the positive class for 80% of the search criteria used to collect training data. This therefore meant that some of the negative training data collected was left out during the process of training the classifier so as to create an equal qualitative benchmark for both classes during computation of performance metrics.

2. Practical Comparison of Models

Time constraints also made it a challenge to be able to implement various classifiers and compare performance with the test data collected. As stated earlier, comparison of other methods and models was left to theory as opposed to practical implementations that may have yielded some new knowledge with regard to classification of sentiment information from tweets.

3. Testing Kenyan Tweets

One focal point of this research was to ensure its "Kenyan-ness". This means the classifier was able to classify tweets written with English, Swahili or a combination of both. In order to test this, the approach adopted was to use already existing classifiers and compare their classification of tweets with Swahili words, to the classification made by the classifier built during this project. However that proved challenging as most existing classifiers made no attempt to classify tweets with phrases in Swahili. In fact, most of these classifiers only returned results of tweets they were able to classify with 100% certainty. It was therefore difficult to asses whether out classifier made any considerable value addition with regards to classification of multilingual tweets.

C. Future Work

There are many further areas of research in this field. Looking at sentiment analysis in general, the integration of other social networks such as facebook, video sharing sites such as youtube and blogs would be an ideal place to begin. This would allow users to use a single web based portal and from it gain access to summarized sentiment scores of opinions expressed in different forms about various topics, all over the internet. Closer to the problem of classifying tweets, further research could be carried out on determining the best learning model to use, and which feature extraction method it would work best with. SVM's and ME classifiers could be discussed with more feature extraction methods such as part of speech tags being included in the research.

VII. APPENDICIES

A. BIBLIOGRAPHY

Accuracy and Precision, 2011. *Accuracy and Precision.* [online] (Updated 5 Jan 2011) Available at: < http://en.wikipedia.org/wiki/Accuracy_and_precision#Accuracy_and_precision_in_binary_classification> [Accessed 17 January 2011]

Benamara, F. Cesarano, C. Picariello, A. Reforgiato, D. and Subrahmanian, V. (2007) .Sentiment analysis: Adjectives and adverbs are better than adjectives alone. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM).*

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

Das, S. and Chen, M., (2001). Yahoo! for Amazon: Extracting market sentiment from stock message boards. In *Proceedings of the Asia Pacific Finance Association Annual Conference (APFA).*

Go, A. B, Richa. and Lei, H. (2009). *Twitter Sentiment Classification using Distant Supervision,* Stanford University, Stanford, CA, USA.

Huffaker, D. A., and Calvert, S. L. (2005). Gender, identity, and language use in teenage blogs. *Journal of Computer-Mediated Communication,* Vol. 10, no. 2, article 1.

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. and 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)*, pages 79–86.

Pang, B. and Lee, L. (2008). Opinion Mining and Sentiment Analysis. *Foundation and Trends in Information Retrieval,* Vol. 2, No. 1-2, pp. 1-135.

Perkins Jacob, 2010. Text Classification for Sentiment Analysis – Naive Bayes Classifier. *StreamHacker*, [blog] May 10, Available at: <http://streamhacker.com/2010/05/10/text-classification-sentiment-analysis-naive-bayes-classifier/> [Accessed 10 January 2011]

Perkins Jacob, 2010. Text Classification for Sentiment Analysis – Precision and Recall. *StreamHacker*, [blog] May 17, Available at: <http://streamhacker.com/2010/05/10/text-classification-sentiment-analysis-precision-recall/> [Accessed 11 January 2011]

Perkins Jacob, 2010. Text Classification for Sentiment Analysis – Stopwords and Collocations. *StreamHacker*, [blog] May 24, Available at: <http://streamhacker.com/2010/05/10/text-classification-sentiment-analysis-stopwords-collocations/> [Accessed 11 January 2011]

Read, J. (2005). Using Emoticons to reduce Dependency in Machine Learning Techniques for Sentiment Classification. In *Proceedings of ACL-05, 43rd Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Sensitivity and Specificity, 2011. *Sensitivity and Specificity*. [online] (Updated 13 Jan 2011) Available at: < http://en.wikipedia.org/wiki/Sensitivity_and_specificity > [Accessed 17 January 2011]

Twitter Sentiment, 2010. *Twitter Sentiment*. [online] Available at: <http://twittersentiment.appspot.com/search?query=google> [Accessed 17 January 2011]

Twitter Sentiment, 2010. *For Researchers – Twitter Sentiment Help*. [online] Available at: < https://sites.google.com/site/twittersentimenthelp/for-researchers> [Accessed 12 January 2011]

Yang, K. Yu, N. Valerio, A. Zhang, H. and Ke, W. (2006). Fusion Approach to Finding opinions in Blogosphere. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)* .

Zhang, W. Meng, W. and Yu, C. (2007) Opinion retrieval from blogs. In *Proceedings of the ACM SIGIR Conference on Information and Knowledge Management (CIKM).*
.Perkins, J., 2010. *Python Text Processing with NLTK 2.0 Cookbook.* 1st ed. Packt Publishing.

B. CODE LISTINGS

1. Code Listing For the Twitter Search Tool

```
<html><head><title> TWITTER SEARCH </title></head>
<body>
<form action="test6.php" method="get">
<input name="twitterq" type="text" id="twitterq" />
<input name="Search" type="submit" value="Search" />
</form>

<?php
header('Content-type: text/html; charset=utf-8');
if (isset($_GET['twitterq']))
  {
   $twitter_query = $_GET['twitterq'] ;

          //print "<h3> Query = $twitter_query </h3>";

require_once 'includes/class.twittersearch.php';
$cnt = 0;

for ($i = 1; $i<= 15; $i++)
   {
$search = new TwitterSearch($twitter_query);

$search->since_date     ='2011-02-20';
$search->rpp                 = 100;
$search->page               = $i;


$data = $search->results();
//$file = fopen('orange_'.date("jSFY").'.txt', 'a');
//$file = fopen($twitter_query.'_'.date("jSFY").'.txt', 'a');
//$file = fopen($twitter_query.'_'.date("jSFY").'.txt', 'a');
$file = fopen($twitter_query.'_'.date("jSFY").'.txt', 'a');

//$data = $s->search('#carsonified');
//$data = $data->results;
?>
<table border=0 cellpadding=4 cellspacing=0 width=80% align=center>



<?php foreach($data as $d){ ?>

      <?php
       // values
                           $cnt ++;

       $rem = $cnt % 2;
```

```
        if ($rem == 0) { $bgc = "#eeeeee"; } else {$bgc = "#ffffff";}
        $image    = $d->profile_image_url ;
        $date     = strtotime($d->created_at) + 10800;
        $author   = $d->from_user ;
        $location = $d->location ;
        $content  = $d->text ;

        //$info     = array($author,$location,$content,$date);
                        //$info     = array($author,$location,$content,date("jS F Y h:i:s A",$date));
        //$info     = array($content);
                        $info     = array($content,date("jS F Y h:i:s A",$date));
                        $line = implode(";;", $info)."\n";
        fwrite($file, $line);

        ?>

         <tr <?php print "bgcolor=$bgc"; ?>>
                <td valign=top><b> <?php echo $cnt  ?></td>
                <td valign=top><img src="<?php //echo $image; ?>" alt="" / ></td>
                <td valign=top><?php //print "<a href=http://twitter.com/$author> $author</a> <br>
$location " ;?></td>
                <td valign=top><?php echo preg_replace('/(^|\s)@(\w+)/','\1<a
href="http://twitter.com/\2">@\2</a>', $content); ?></td>
                <td valign=top><?php print date("jS F Y h:i:s A",$date) ; ?></td>
    </tr>

<?php } ?>
</table>

<?php }?>
<?php }?>
```

2. Code Listing For the Sentiment Analysis Tool

```
<html>
<head>
<title> TWITTER SENTIMENT ANALYSER </title></head>
<body>
<div style="position:relative; left: 205; width: 572;">
<H2>TWITTER SENTIMENT ANALYSIS</H2></div>
<form action="test14.php" method="get">

  <div style="position:relative; left: 205; width: 572;">
    <input name="twitterq" type="text" id="twitterq" size="30"/>
    <input name="Search" type="submit" value="Search" />
  </div>
</form>
<link href="tweets.css" rel="stylesheet" type="text/css" />


<?php
header('Content-type: text/html; charset=utf-8');
if (isset($_GET['twitterq']))
   {
    $twitter_query = $_GET['twitterq'] ;
?>

<div style = "font-size:12px; border:solid 1px #FFFFFF; position:relative; left: 205; width: 572;">
<H2>Showing Results For <?php echo $twitter_query; ?></H2>
</div>


<?php
require_once 'includes/class.twittersearch.php';

$cnt = 0;
$tpos = 0;
$tneg = 0;
$tneu = 0;
$tsents = 0;
$j = 1;
$k=0;
$daily_neg = 0;
$daily_pos = 0;
$daily_neu = 0;


//$rset = array();
//$row = array();

$initdata = 0;
$resultdata=array();


for ($a = 0; $a<5; $a++) {
            $resultdata[$a]=array();
            for ($b = 0; $b<3; $b++) {
```

```
                        $resultdata[$a][$b] = $initdata;
                }
}


$today = date("Y-m-d");
$newdate = strtotime ( $today . ' -4 day' ) ;
$sincedate = date ( 'Y-m-d' , $newdate );

for ($i = 1; $i<= 1; $i++)
   {
$search = new TwitterSearch($twitter_query);
$search->since_date  = $sincedate;
$search->rpp    = 20;
$search->page = $i;


$data = $search->results();
$first = $data[0];
$cdate = strtotime($first->created_at) + 10800;
$current_date = date("Y-m-d",$cdate);
//echo $current_date;
//$today = mktime(0, 0, 0, date('m'), date('d'), date('Y'));
//$current_date = $today


?>

<?php foreach($data as $d) { ?>


      <?php

                              $store = 'workfile.txt';

                              if ( !file_exists($store)){
                                      touch ($store);
                                      $file = fopen ($store, 'w');

                              }
                              else{
                              $file = fopen ($store, 'w');
                              }

                              $cnt ++;
                              $image    = $d->profile_image_url ;
      $date     = strtotime($d->created_at).+ 10800;
                              $timest    =        $date * 1000;
      $author   = $d->from_user ;
      $location = $d->location ;
      $content  = $d->text ;


                              $check_date = date("Y-m-d",$date);
                              $new_date = date ("Y, m, d",$timest);
```

```
                    //while next($date) = $check_date {
                    //while $check_date = $current_date {

                    if (strtotime($check_date) == strtotime($current_date)) {


                    }
                    else{

                            //$subtract = $current_date - 86400;
                            //$current_date = date ("Y-m-d", $subtract);
                            $current_date = $check_date;
                            $j++;
                            $k++;
                            $daily_neg = 0;
                            $daily_pos = 0;
                            $daily_neu = 0;

                    }

                    $info = array($content);
                    $line = implode(" ; ", $info)."\n";
                    fwrite($file, $line);

                    exec("C:\\Python26\\python.exe C:\\wamp\\www\\tweets\\tweets9.py");

                    $output = 'output.txt';
                    $fh = fopen($output, 'r');
                    $sent = fgets($fh);
                    $d->sentiment = $sent;


                    fwrite($file, "");
                    fclose($file);
                    fclose($fh);

                    if ($sent == 'neg') {$tneg ++; $daily_neg ++;} elseif ($sent == 'pos'){$tpos ++;
$daily_pos ++;} else {$tneu ++; $daily_neu ++;}

                    //$rset = array("day_".$j, $daily_neg, $daily_pos);

                    $store_date = date("Y, n, j",$date);
                    $timestamp = (strtotime($check_date) - 10800) * 1000;

                    $resultdata[$k][0] = $timestamp;
                    $resultdata[$k][1] = $daily_pos;
                    $resultdata[$k][2] = $daily_neg;
                    $resultdata[$k][3] = $daily_neu;

                    //echo '|'.$resultdata[0][0].'|'.$resultdata[0][1].'|'.$resultdata[0][2].'|<br />';


    ?>

<?php } ?>
<?php
```

```
//echo '|'.$resultdata[0][0].'|'.$resultdata[0][1].'|'.$resultdata[0][2].'|<br />';
//echo '|'.$resultdata[1][0].'|'.$resultdata[1][1].'|'.$resultdata[1][2].'|<br />';
//echo '|'.$resultdata[2][0].'|'.$resultdata[2][1].'|'.$resultdata[2][2].'|<br />';
//echo '|'.$resultdata[3][0].'|'.$resultdata[3][1].'|'.$resultdata[3][2].'|<br />';
//echo '|'.$resultdata[4][0].'|'.$resultdata[4][1].'|'.$resultdata[4][2].'|<br />';

?>
<?php }?>

<?php
require ('includes/GChartPhp/GChartPhp/gChart.php');
?>
<?php
$tsents = $tneg + $tpos + $tneu;

$pos_percent = (($tpos/$cnt) * 100);
$neg_percent = (($tneg/$cnt) * 100);
$neu_percent = (($tneu/$cnt) * 100);

$piChart = new gPieChart();
$piChart->addDataSet(array($pos_percent,$neg_percent,$neu_percent));
$piChart->setLegend(array("positive", "negative", "neutral"));
$piChart->setLabels(array($pos_percent."%", $neg_percent."%", $neu_percent."%"));
$piChart->setColors(array("AFDCEC", "FAAFBA", "D8D8D8"));
$piChart->setRotation("130");



$barChart = new gBarChart(350,200,'g','v');
$barChart->addDataSet(array($tpos));
$barChart->addDataSet(array($tneg));
$barChart->addDataSet(array($tneu));
$barChart->setColors(array("AFDCEC", "FAAFBA", "D8D8D8"));
$barChart->setDataRange(0, $tsents);
$barChart->setLegend(array("pos = ".$tpos, "neg = ".$tneg, "neu = ".$tneu));
$barChart->setVisibleAxes(array('y'));
$barChart->addAxisRange(0,0,$tsents);
$barChart->setAutoBarWidth();

//$barChart->setBarWidth(17,15);

?>


<?php

include_once 'includes/qgooglevisualapi/config.inc.php';

$chart = new QAnnotatedtimelineGoogleGraph();
$chart
            //->ignoreContainer()
            ->addDrawProperties(
                  array(
                        "title"=>'Sentiment Timeline',
                        )
```

```
                    )
            ->addColumns(
                    array(
                array('date', 'Date'),
                array('number', 'Positive'),
                array('number', 'Negative'),
                            array('number', 'Neutral'),
                            )
                    );
for ($i= 0; $i<= $k; $i++)
{
$chart
            ->setValues(
                    array(
                            array($i, 0, 'new Date('.$resultdata[$i][0].')'),
                            array($i, 1, $resultdata[$i][1]),
                            array($i, 2, $resultdata[$i][2]),
                            array($i, 3, $resultdata[$i][3]),
                    )
            );

}
//echo $chart->getReferenceLink();
?>

<div>
        <table border=0 cellpadding=1 cellspacing=6 width=80% align=center>
        <tr>
         <td align=center><H3>Sentiment Analysis Results By Percantage</H3></td>
                <td align=center><H3>Sentiment Analysis Results By Number</H3></td>
        </tr>
        <tr>
                <td align=center><img src="<?php print $piChart->getUrl();  ?>"/></td>
                <td align=right><img src="<?php print $barChart->getUrl(); ?>"/></td>
        </tr>
 </table>
</div>

<div>
        <table border=0 cellpadding=1 cellspacing=6 width=80% align=center>
        <tr>
                <td align=center><H3>Sentiment Analysis Timeline</H3></td>
        </tr>
        <tr>
                <td align=center><?php echo $chart->render();?></td>
        </tr>
 </table>

</div>

<div style = "font-size:15px; border:solid 1px #FFFFFF; position:relative; left: 85; width: 572;">
<H3>Tweets Containing <?php echo $twitter_query; ?></H3>
</div>

<div class ="twitter_container">
<?php //foreach($data as $d){ ?>
```

```
<?php
$end = 0;
if ($cnt > 20) {$end = 20;} else {$end = $cnt;}
?>

<?php for ($i= 0; $i< $end; $i++){ ?>

                <?php
                        //for ($i= $cnt; $i>= $cnt - 3; $i=$i-1){
                        $d          =       $data[$i];
                        $image   = $d->profile_image_url ;
        $date     = strtotime($d->created_at) + 10800;
        $author   = $d->from_user ;
        $location = $d->location ;
        $content  = $d->text ;
                        $sentdata = $d->sentiment;

                        if ($sentdata == 'neg') { $bgc = "negative";} elseif ($sentdata == 'pos') { $bgc =
"positive";} else {$bgc = "neutral";}

                        echo '<div class = "'.$bgc.'">';
                        echo '<div class="twitter_status">';
                        echo $content;
                        echo '<div class="twitter_small">';
                        echo '<strong>Posted On:</strong>';
                        echo date("jS F Y h:i:s A",$date);
                        echo '</div>';
                        echo '</div>';
                        echo '</div>';
                        //}
                ?>
<?php }?>
</div>
<?php }?>

</body>
</html>
```

3. **Reused Code Listing For the Twitter API Wrapper**

```
<?php
/**
 * Wrapper class around the Twitter Search API for PHP
 * Based on the class originally developed by David Billingham
 * and accessible at http://twitter.slawcup.com/twitter.class.phps
 * @author Ryan Faerman <ryan.faerman@gmail.com>
 * @version 0.2
 * @package PHPTwitterSearch
 */
class TwitterSearch {
        /**
         * Can be set to JSON (requires PHP 5.2 or the json pecl module) or XML - json|xml
         * @var string
         */
	var $type = 'json';

	/**
	 * It is unclear if Twitter header preferences are standardized, but I would suggest using them.
	 * More discussion at http://tinyurl.com/3xtx66
	 * @var array
	 */
	var $headers=array('X-Twitter-Client: PHPTwitterSearch','X-Twitter-Client-Version: 0.1','X-
Twitter-Client-URL: http://ryanfaerman.com/twittersearch');

	/**
	 * Recommend setting a user-agent so Twitter knows how to contact you inc case of abuse.
Include your email
	 * @var string
	 */
	var $user_agent='';

	/**
	 * @var string
	 */
	var $query='';

	/**
	 * @var array
	 */
	var $responseInfo=array();

	/**
	 * Use an ISO language code. en, de...
	 * @var string
	 */
	var $lang;

	/**
	 * The number of tweets to return per page, max 100
	 * @var int
	 */
	var $rpp;

	/**
	 * The page number to return, up to a max of roughly 1500 results
	 * @var int
	 */
	var $page;

	/**
	 * Return tweets with a status id greater than the since value
	 * @var int
	 */
	var $since;

	/**
	 * Returns tweets by users located within a given radius of the given latitude/longitude, where the
user's location is taken from their Twitter profile. The parameter value is specified by
"latitide,longitude,radius", where radius units must be specified as either "mi" (miles) or "km" (kilometers)
	 * @var string
	 */
	var $geocode;

	/**
	 * When "true", adds "<user>:" to the beginning of the tweet. This is useful for readers that do not
display Atom's author field. The default is "false"
	 * @var boolean
	 */
	var $show_user = false;


	var $since_date;


	var $until_date;


	/**
	* @param string $query optional
	*/
	function TwitterSearch($query=false) {
		$this->query = $query;
	}

	/**
	* Find tweets from a user
	* @param string $user required
	* @return object
	*/
	function from($user) {
		$this->query .= ' from:'.str_replace('@', '', $user);
		return $this;
	}

	/**
	* Find tweets to a user
	* @param string $user required
	* @return object
	*/
	function to($user) {
		$this->query .= ' to:'.str_replace('@', '', $user);
		return $this;
	}

	/**
	* Find tweets referencing a user
	* @param string $user required
	* @return object
	*/
	function about($user) {
		$this->query .= ' @'.str_replace('@', '', $user);
		return $this;
	}

	/**
	* Find tweets containing a hashtag
* @param string $user required
* @return object
*/
function with($hashtag) {
        $this->query .= ' #'.str_replace('#', '', $hashtag);
        return $this;
}

/**
* Find tweets containing a word
* @param string $user required
* @return object
*/
function contains($word) {
        $this->query .= ' '.$word;
        return $this;
}

/**
* Set show_user to true
* @return object
*/
function show_user() {
        $this->show_user = true;
        return $this;
}

/**
* @param int $since_id required
* @return object
*/
function since($since_id) {
        $this->since = $since_id;
        return $this;
}

/**
* @param int $language required
* @return object
*/
function lang($language) {
        $this->lang = $language;
        return $this;
}

/**
* @param int $n required
* @return object
*/
function rpp($n) {
        $this->rpp = $n;
        return $this;
}

/**
* @param int $n required
* @return object
*/
function page($n) {
        $this->page = $n;
        return $this;
}

function since_date($since_date) {
        $this->since_date = $since_date;
        return $this;
}

function until_date($until_date) {
        $this->until_date = $until_date;
        return $this;
}


/**
* @param float $lat required. lattitude
* @param float $long required. longitude
* @param int $radius required.
* @param string optional. mi|km
* @return object
*/
function geocode($lat, $long, $radius, $units='mi')
{

        $this->geocode = $lat.','.$long.','.$radius.$units;
//      $this->geocode = $lat.','.$long.','.$radius.','.$units;
        return $this;
}

/**
* Build and perform the query, return the results.
* @param $reset_query boolean optional.
* @return object
*/
function results($reset_query=true) {
        //$request  = 'http://search.twitter.com/search.'.$this->type;
        //$request  = 'http://search.twitter.com/search.'.$this->type;
        $request  = 'http://search.twitter.com/search.'.$this->type;
        $request .= '?geocode=-1.274359%2C36.813106%2C15.0mi&';
        $request .= 'q='.urlencode($this->query);
        //$request .= '&since=2011-01-21';


        if(isset($this->since_date)) {
                $request .= '&since='.$this->since_date;
        }

        if(isset($this->until_date)) {
                $request .= '&until='.$this->until_date;
        }

        if(isset($this->rpp)) {
```

```
			$request .= '&rpp='.$this->rpp;
		}

		if(isset($this->page)) {
			$request .= '&page='.$this->page;
		}

		if(isset($this->lang)) {
			$request .= '&lang='.$this->lang;
		}

		if(isset($this->since)) {
			$request .= '&since_id='.$this->since;
		}

		if($this->show_user) {
			$request .= '&show_user=true';
		}

		if(isset($this->geocode)) {
			print $this->geocode ;
			$request .= '&geocode='.$this->geocode;
		}

		if($reset_query) {
			$this->query = '';
		}

		//print $request;

		return $this->objectify($this->process($request))->results;
	}

	/**
	* Returns the top ten queries that are currently trending on Twitter.
	* @return object
	*/
	function trends() {
		$request  = 'http://search.twitter.com/trends.json';

		return $this->objectify($this->process($request));
	}

	/**
	 * Internal function where all the juicy curl fun takes place
	 * this should not be called by anything external unless you are
	 * doing something else completely then knock youself out.
	 * @access private
	 * @param string $url Required. API URL to request
	 * @param string $postargs Optional. Urlencoded query string to append to the $url
	 */
	function process($url, $postargs=false) {
		$ch = curl_init($url);
		if($postargs !== false) {
			curl_setopt ($ch, CURLOPT_POST, true);
			curl_setopt ($ch, CURLOPT_POSTFIELDS, $postargs);
	}




	curl_setopt($ch, CURLOPT_VERBOSE, 1);
	curl_setopt($ch, CURLOPT_NOBODY, 0);
	curl_setopt($ch, CURLOPT_HEADER, 0);
	curl_setopt($ch, CURLOPT_USERAGENT, $this->user_agent);
	curl_setopt($ch, CURLOPT_FOLLOWLOCATION,1);
	curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
	curl_setopt($ch, CURLOPT_HTTPHEADER, $this->headers);
			curl_setopt($ch, CURLOPT_PROXY, "192.168.4.4:8080");
			//curl_setopt($ch, CURLOPT_PROXY, "10.2.21.98:80");
			//curl_setopt($ch, CURLOPT_PROXYUSERPWD, "[uhs]:[uhs]");
	$response = curl_exec($ch);

	$this->responseInfo=curl_getinfo($ch);
	curl_close($ch);

	if( intval( $this->responseInfo['http_code'] ) == 200 )
				return $response;
	else
	   return false;
	  }

	  /**
	   * Function to prepare data for return to client
	   * @access private
	   * @param string $data
	   */
	  function objectify($data) {
			if( $this->type ==  'json' )
				return (object) json_decode($data);

			else if( $this->type == 'xml' ) {
				if( function_exists('simplexml_load_string') ) {
					$obj = simplexml_load_string( $data );

					$statuses = array();
					foreach( $obj->status as $status ) {
						$statuses[] = $status;
					}
					return (object) $statuses;
				}
				else {
					return $out;
				}
			}
			else
				return false;
	  }
}

?>
```

4. Code Listing For the Learning Model

```
from nltk.corpus import CategorizedPlaintextCorpusReader
import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
from nltk.util import bigrams
from nltk.corpus import stopwords
import nltk.classify.util, nltk.metrics
from nltk.metrics import BigramAssocMeasures
from nltk.probability import FreqDist, ConditionalFreqDist
import collections, itertools
import cPickle as pickle

corpus_root = r'C:\nltk_data\corpora\KWTweets'
file_pattern = r'.*\.txt'
tweets = CategorizedPlaintextCorpusReader(corpus_root,file_pattern,cat_pattern= r'(\w+)')

test_corpus_root = r'C:\nltk_data\corpora\HLTweets'
test_file_pattern = r'.*\.txt'
test_tweets = CategorizedPlaintextCorpusReader(test_corpus_root,test_file_pattern,cat_pattern= r'(\w+)')


def good_word_feats(words):
                    return dict([(word, True) for word in words if not(word.lower().strip(",").strip("!") in ["r",
"u", "ur", "urs", "we", "was", "2", "at", "that", "it", "for", "is", "i", "on", "of", "am", "i'm", ":", ".", ";", "!",
"&", ",", "...", "www","&quot:", "?",",", "--", "&gt;", "&gt;&gt;", "'", "in", "the", "!!", "+", "!!!", ":p",
"&gt", "&gt:", ":o", ";o", "&lt;3", "a", "rt"]) and (not word.lower() in stopwords.words('english')) and (not
word.lower().isdigit()) and (not word.lower().startswith("#")) and (not word.lower().startswith("@")) and
(not word.startswith("RT")) and (not word.lower().startswith("http"))])

def good_words(words):
                    return [(word) for word in words if not(word.lower().strip(",").strip("!") in ["r", "u", "ur",
"urs", "we", "was", "2", "at", "that", "it", "for", "is", "i", "on", "of", "am", "i'm", ":", ".", ";", "!", "&", ",",
"...", "www","&quot;", "?",",", "--", "&gt;", "&gt;&gt;", "'", "in", "the", "!!", "+", "!!!", ":p", "&gt", "&gt:",
":o", ";o", "&lt;3", "a", "rt"]) and (not word.lower() in stopwords.words('english')) and (not
word.lower().isdigit()) and (not word.lower().startswith("#")) and (not word.lower().startswith("@")) and
(not word.startswith("RT")) and (not word.lower().startswith("http"))]

def get_words(words):
                    return [(word) for word in words]

def bigram_word_feats(words):

                    mybigrams = bigrams(words)
                    return dict([(bigram, True) for bigram in mybigrams])

def good_bigram_word_feats(words):

                    mybigrams = bigrams(good_words(words))
                    return dict([(bigram, True) for bigram in mybigrams])


def bow_word_feats(words):
```

```
                    return dict([(word, True) for word in words])


def comb_bigram_word_feats(words):

        mybigrams = bigrams(words)
        d = dict([(bigram, True) for bigram in mybigrams])
        d.update(bow_word_feats(words))
        return d

def comb_good_bigram_word_feats(words):

        mybigrams = bigrams(good_words(words))
        d = dict([(bigram, True) for bigram in mybigrams])
        d.update(good_word_feats(words))
        return d

def clean_up(words):
        return [word.lower().lstrip('#').lstrip('@').rstrip('!').strip('*') for word in words]


negids = tweets.fileids(categories=['neg'])
posids = tweets.fileids(categories=['pos'])

neg_test_ids = test_tweets.fileids(categories=['neg'])
pos_test_ids = test_tweets.fileids(categories=['pos'])


negwords = [(good_words(clean_up(str.split(tweets.raw(fileids=[f]))))) for f in negids]
poswords = [(good_words(clean_up(str.split(tweets.raw(fileids=[f]))))) for f in posids]


negfeats = [(comb_good_bigram_word_feats(clean_up(str.split(tweets.raw(fileids=[f])))), 'neg') for f in
negids]
posfeats = [(comb_good_bigram_word_feats(clean_up(str.split(tweets.raw(fileids=[f])))), 'pos') for f in
posids]

neg_test_feats = [(comb_good_bigram_word_feats(clean_up(str.split(test_tweets.raw(fileids=[f])))), 'neg')
for f in neg_test_ids]
pos_test_feats = [(comb_good_bigram_word_feats(clean_up(str.split(test_tweets.raw(fileids=[f])))), 'pos')
for f in pos_test_ids]


negcutoff = len(neg_test_feats)*5/8
poscutoff = len(pos_test_feats)*5/8

trainfeats = negfeats + posfeats
testfeats = neg_test_feats[negcutoff:] + pos_test_feats[poscutoff:]
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)

print 'accuracy:', nltk.classify.util.accuracy(classifier, testfeats)
```

```
classifier.show_most_informative_features()

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 'neg precision:', nltk.metrics.precision(refsets['neg'], testsets['neg'])
print 'neg recall:', nltk.metrics.recall(refsets['neg'], testsets['neg'])



filename = open('KWTD_NB_Classifier.pickle', 'wb')
pickle.dump(classifier, filename)
filename.close()
```

5. Code Listing for the Live Data Sentiment Classifier Model

```
from nltk.corpus import CategorizedPlaintextCorpusReader
import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
from nltk.util import bigrams
from nltk.corpus import stopwords
import nltk.classify.util, nltk.metrics
from nltk.metrics import BigramAssocMeasures
from nltk.probability import FreqDist, ConditionalFreqDist
import collections, itertools
import cPickle as pickle



def good_word_feats(words):
                    return dict([(word, True) for word in words if not(word.lower().strip(",").strip("!") in ["r",
"u", "ur", "urs", "we", "was", "2", "at", "that", "it", "for", "is", "i", "on", "of", "am", "i'm", ":", ".", ";", "!",
"&", ",", "...", "www","&quot;", "?",",", "--", "&gt;", "&gt;&gt;", ";)", ":'(", ":/", ";\\", "'", "=d", "in", "the",
":d", "!!", "+", "!!!", ":p", "&gt", "&gt:", ":o", ";o", "&lt;3", "a", "rt"]) and (not word.lower() in
stopwords.words('english')) and (not word.lower().isdigit()) and (not word.lower().startswith("#")) and (not
word.lower().startswith("@")) and (not word.startswith("RT")) and (not word.lower().startswith("http"))])

def good_words(words):
                    return [(word) for word in words if not(word.lower().strip(",").strip("!") in ["r", "u", "ur",
"urs", "we", "was", "2", "at", "that", "it", "for", "is", "i", "on", "of", "am", "i'm", ":D", ":P", "D", "P", ":",
".", ";", "!", "&", ",", "...", ":)", ":(", ":-)", ":-(", ")", "(", "=", "=)", "^_^", "-", "$lt;3", ":/", ":|",
"www","&quot;", "?",",", "--", "&gt;", "&gt;&gt;", ";)",":'(", ":/", ";\\", "'", "=d", "in", "the", ":d", "!!", "+",
"!!!", ":p", "&gt", "&gt:", ":o", ";o", "&lt;3", "a", "rt"]) and (not word.lower() in
stopwords.words('english')) and (not word.lower().isdigit()) and (not word.lower().startswith("#")) and (not
word.lower().startswith("@")) and (not word.startswith("RT")) and (not word.lower().startswith("http"))]

def words(words):
                    return [(word) for word in words]
```

```
def bigram_word_feats(words):
                    mybigrams = bigrams(words)
                    return dict([(bigram, True) for bigram in mybigrams])

def good_bigram_word_feats(words):
                    mybigrams = bigrams(good_words(words))
                    return dict([(bigram, True) for bigram in mybigrams])

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

def comb_good_bigram_word_feats(words):
          mybigrams = bigrams(words)
          d = dict([(bigram, True) for bigram in mybigrams])
          d.update(good_word_feats(words))
          return d

def clean_up(words):

          return [word.lower().lstrip('#').lstrip('@').rstrip('!').rstrip('!!!').rstrip('...').strip('*').strip(',') for word
in words]

def tweetsent(words):
          value = 0
          for i in words:
                    if [i] in negwords:
                                                  value += 1
                    elif [i] in poswords:
                                                  value += 1
                    else:
                                                  value += 0
          return value


def biextract(words):
          bivalue = 0
          for a,b in words:
                              if [a,b] in negwords:
                                                  bivalue += 1
                              elif [a,b] in poswords:
                                                  bivalue += 1
                              else:
                                                  bivalue += 0
          return bivalue


items = ["?", "!", ".", "", ","]


corpus_root = r'C:\nltk_data\corpora\KWTweets'
file_pattern = r'.*\.txt'
tweets = CategorizedPlaintextCorpusReader(corpus_root,file_pattern,cat_pattern= r'(\w+)')

negids = tweets.fileids(categories=['neg'])
posids = tweets.fileids(categories=['pos'])
```

```
negwords = [(good_words(clean_up(str.split(tweets.raw(fileids=[f]))))) for f in negids]
poswords = [(good_words(clean_up(str.split(tweets.raw(fileids=[f]))))) for f in posids]


data = open('workfile.txt', 'r')
tweet= data.readline()
s = list(tweet)
for i in items:
   s = [s.replace(i, ' ') for s in s]

newtweet ="".join(s)
classifier = pickle.load(open('KWTD_NB_Classifier.pickle'))

tweetfeats = (good_words(clean_up(str.split(newtweet))))
bifeats = (bigrams(good_words(clean_up(str.split(newtweet)))))

k = 0
newbifeats= []
for i in bifeats:
   var = list(bifeats[k])
   newbifeats.append(var)
   k += 1

wordresults = tweetsent(tweetfeats)
bifeatresults = biextract(newbifeats)
tresults = wordresults + bifeatresults


if tresults == 0:
        sent = 'neu'
else:
        sent = classifier.classify(bow_word_feats(clean_up(str.split(newtweet))))


results = open('output.txt', 'w')
results.write(sent)
```

C. DETAILED TEST RESULTS

1. USE OF EMOTICON LABELED TRAINING DATA AND HAND LABELED TEST DATA

a) UNIGRAMS

(1) With Noisy Data

```
train on 5000 instances, test on 300 instances
accuracy: 0.693333333333
```

```
Most Informative Features
                :-( = True          neg : pos    =    107.0 : 1.0
                 :( = True          neg : pos    =     63.5 : 1.0
                 :p = True          neg : pos    =     34.0 : 1.0
                :-) = True          pos : neg    =     17.6 : 1.0
                 :d = True          pos : neg    =     14.0 : 1.0
                 :) = True          pos : neg    =     13.7 : 1.0
          @noninie: = True          pos : neg    =     11.0 : 1.0
         @hotlivia: = True          neg : pos    =     10.3 : 1.0
                hme = True          neg : pos    =      9.7 : 1.0
               4nai = True          neg : pos    =      9.7 : 1.0
pos precision: 0.701388888889
pos recall: 0.673333333333
neg precision: 0.685897435897
neg recall: 0.713333333333
```

(2) Without Noisy Data

```
train on 5000 instances, test on 300 instances
accuracy: 0.703333333333
Most Informative Features
               4nai = True          neg : pos    =      9.7 : 1.0
               2pak = True          neg : pos    =      9.7 : 1.0
                hme = True          neg : pos    =      9.7 : 1.0
                 er = True          neg : pos    =      9.0 : 1.0
             lovely = True          pos : neg    =      8.6 : 1.0
            confirm = True          neg : pos    =      8.3 : 1.0
               city = True          neg : pos    =      8.3 : 1.0
               pain = True          neg : pos    =      8.3 : 1.0
            believe = True          neg : pos    =      7.8 : 1.0
            website = True          neg : pos    =      7.7 : 1.0
pos precision: 0.676300578035
pos recall: 0.78
neg precision: 0.740157480315
neg recall: 0.626666666667
```

b) BIGRAMS

(1) With Noisy Data

```
train on 5000 instances, test on 300 instances
accuracy: 0.57
Most Informative Features
      ('thank', 'you') = True          pos : neg    =     23.7 : 1.0
          (':(', ':(') = True          neg : pos    =     19.0 : 1.0
           (':(', 'i') = True          neg : pos    =     13.8 : 1.0
        (':(', 'miss') = True          neg : pos    =     12.3 : 1.0
          ('me', ':)') = True          pos : neg    =     12.3 : 1.0
        ('come', 'to') = True          neg : pos    =     11.7 : 1.0
       ('miss', 'msa') = True          neg : pos    =     11.0 : 1.0
   ('rt', '@noninie:') = True          pos : neg    =     11.0 : 1.0
```

```
        ('it', ':(') = True             neg : pos    =     11.0 : 1.0
   ('rt', '@hotlivia:') = True              neg : pos    =     10.3 : 1.0
pos precision: 0.551724137931
pos recall: 0.746666666667
neg precision: 0.60824742268
neg recall: 0.393333333333
```

(2) Without Noisy Data

```
train on 5000 instances, test on 300 instances
accuracy: 0.5
Most Informative Features
        ('miss', 'msa') = True            neg : pos    =     11.0 : 1.0
      ('headed', 'hme') = True              neg : pos    =      9.7 : 1.0
       ('2pak', '4nai') = True            neg : pos    =      9.7 : 1.0
       ('4nai', 'miss') = True            neg : pos    =      9.7 : 1.0
        ('hme', '2pak') = True             neg : pos    =      9.7 : 1.0
   ('happy', 'birthday') = True             pos : neg    =      7.7 : 1.0
  ('looking', 'forward') = True              neg : pos    =      7.0 : 1.0
    ('chatting', 'pops') = True             neg : pos    =      6.3 : 1.0
       ("didn't", 'get') = True            neg : pos    =      6.3 : 1.0
('msa', 'alrdy!&lt;&gt;msa') = True              neg : pos    =      5.7 : 1.0
pos precision: 0.5
pos recall: 0.933333333333
neg precision: 0.5
neg recall: 0.0666666666667
```

c) COMBINED UNIGRAMS AND BIGRAMS

(1) With Noisy Data

```
train on 5000 instances, test on 300 instances
accuracy: 0.686666666667
Most Informative Features
                  :-( = True            neg : pos    =    107.0 : 1.0
                   :( = True            neg : pos    =     63.5 : 1.0
                   :p = True            neg : pos    =     34.0 : 1.0
      ('thank', 'you') = True             pos : neg    =     23.7 : 1.0
         (':(', ':(') = True            neg : pos    =     19.0 : 1.0
                  :-) = True            pos : neg    =     17.6 : 1.0
                   :d = True            pos : neg    =     14.0 : 1.0
          (':(', 'i') = True            neg : pos    =     13.8 : 1.0
                   :) = True            pos : neg    =     13.7 : 1.0
        (':(', 'miss') = True             neg : pos    =     12.3 : 1.0
pos precision: 0.684210526316
pos recall: 0.693333333333
neg precision: 0.689189189189
neg recall: 0.68
```

(2) Without Noisy Data

```
train on 5000 instances, test on 300 instances
accuracy: 0.7
Most Informative Features
        ('miss', 'msa') = True          neg : pos   =   11.0 : 1.0
      ('headed', 'hme') = True          neg : pos   =    9.7 : 1.0
       ('2pak', '4nai') = True          neg : pos   =    9.7 : 1.0
       ('4nai', 'miss') = True          neg : pos   =    9.7 : 1.0
                    hme = True          neg : pos   =    9.7 : 1.0
        ('hme', '2pak') = True          neg : pos   =    9.7 : 1.0
                   4nai = True          neg : pos   =    9.7 : 1.0
                   2pak = True          neg : pos   =    9.7 : 1.0
                     er = True          neg : pos   =    9.0 : 1.0
                 lovely = True          pos : neg   =    8.6 : 1.0
pos precision: 0.678571428571
pos recall: 0.76
neg precision: 0.727272727273
neg recall: 0.64
```

2. USE OF HAND LABELED TRAINING AND TEST DATA

a) UNIGRAMS

(1) With Noisy Data

```
train on 500 instances, test on 300 instances
accuracy: 0.75
Most Informative Features
                awesome = True          pos : neg   =    9.7 : 1.0
                   love = True          pos : neg   =    8.6 : 1.0
                     :( = True          neg : pos   =    8.3 : 1.0
                   hate = True          neg : pos   =    7.7 : 1.0
                  great = True          pos : neg   =    7.7 : 1.0
                   2011 = True          pos : neg   =    7.7 : 1.0
               february = True          pos : neg   =    7.7 : 1.0
                    his = True          pos : neg   =    5.7 : 1.0
                     pm = True          pos : neg   =    5.7 : 1.0
                 museum = True          pos : neg   =    5.0 : 1.0
pos precision: 0.781954887218
pos recall: 0.693333333333
neg precision: 0.724550898204
neg recall: 0.806666666667
```

(2) Without Noisy Data

```
train on 500 instances, test on 300 instances
accuracy: 0.763333333333
Most Informative Features
               awesome = True           pos : neg    =      9.7 : 1.0
                  love = True           pos : neg    =      8.6 : 1.0
                    :( = True           neg : pos    =      8.3 : 1.0
                  hate = True           neg : pos    =      7.7 : 1.0
                 great = True           pos : neg    =      7.7 : 1.0
              february = True           pos : neg    =      7.7 : 1.0
                    pm = True           pos : neg    =      5.7 : 1.0
                museum = True           pos : neg    =      5.0 : 1.0
                  work = True           neg : pos    =      5.0 : 1.0
                  free = True           pos : neg    =      4.3 : 1.0
pos precision: 0.761589403974
pos recall: 0.766666666667
neg precision: 0.765100671141
neg recall: 0.76
```

b) BIGRAMS

(1) With Noisy Data

```
train on 500 instances, test on 300 instances
accuracy: 0.593333333333
Most Informative Features
           ('at', 'the') = True           pos : neg    =      7.7 : 1.0
    ('february', '2011') = True           pos : neg    =      7.7 : 1.0
        ('lebron', 'is') = True           pos : neg    =      5.7 : 1.0
         ('night', 'at') = True           pos : neg    =      5.0 : 1.0
     ('the', 'museum') = True           pos : neg    =      5.0 : 1.0
           ('i', 'hate') = True           neg : pos    =      4.3 : 1.0
             ('i', 'am') = True           pos : neg    =      3.7 : 1.0
            ('up', 'my') = True           neg : pos    =      3.0 : 1.0
         ('with', 'the') = True           neg : pos    =      3.0 : 1.0
          ('the', 'new') = True           pos : neg    =      3.0 : 1.0
pos precision: 0.567307692308
pos recall: 0.786666666667
neg precision: 0.652173913043
neg recall: 0.4
```

(2) Without Noisy Data

```
train on 500 instances, test on 300 instances
accuracy: 0.55
Most Informative Features
     ('night', 'museum') = True           pos : neg    =      5.0 : 1.0
   ('customer', 'care') = True           neg : pos    =      3.0 : 1.0
    ('san', 'francisco') = True           neg : pos    =      2.3 : 1.0
        ('twitter', 'api') = True           neg : pos    =      2.2 : 1.0
 ('malcolm', 'gladwell') = True           pos : neg    =      1.9 : 1.0
```

```
        ('palo', 'alto') = True          pos : neg    =    1.7 : 1.0
        ('east', 'palo') = True          pos : neg    =    1.7 : 1.0
   ('lambda', 'calculus') = True          neg : pos    =    1.7 : 1.0
    ('allen', 'hamilton') = True          pos : neg    =    1.7 : 1.0
        ('booz', 'allen') = True          pos : neg    =    1.7 : 1.0
pos precision: 0.526690391459
pos recall: 0.986666666667
neg precision: 0.894736842105
neg recall: 0.113333333333
```

c) USING BOTH UNIGRAMS AND BIGRAMS

(1) With Noisy Data

```
train on 500 instances, test on 300 instances
accuracy: 0.746666666667
Most Informative Features
               awesome = True          pos : neg    =    9.7 : 1.0
                  love = True          pos : neg    =    8.6 : 1.0
                    :( = True          neg : pos    =    8.3 : 1.0
          ('at', 'the') = True          pos : neg    =    7.7 : 1.0
                  hate = True          neg : pos    =    7.7 : 1.0
                 great = True          pos : neg    =    7.7 : 1.0
                  2011 = True          pos : neg    =    7.7 : 1.0
              february = True          pos : neg    =    7.7 : 1.0
    ('february', '2011') = True          pos : neg    =    7.7 : 1.0
                    pm = True          pos : neg    =    5.7 : 1.0
pos precision: 0.776119402985
pos recall: 0.693333333333
neg precision: 0.722891566265
neg recall: 0.8
```

(2) Without Noisy Data

```
train on 500 instances, test on 300 instances
accuracy: 0.753333333333
Most Informative Features
               awesome = True          pos : neg    =    9.7 : 1.0
                  love = True          pos : neg    =    8.6 : 1.0
                    :( = True          neg : pos    =    8.3 : 1.0
                  hate = True          neg : pos    =    7.7 : 1.0
                 great = True          pos : neg    =    7.7 : 1.0
              february = True          pos : neg    =    7.7 : 1.0
                    pm = True          pos : neg    =    5.7 : 1.0
                museum = True          pos : neg    =    5.0 : 1.0
     ('night', 'museum') = True          pos : neg    =    5.0 : 1.0
                  work = True          neg : pos    =    5.0 : 1.0
```

```
pos precision: 0.75
pos recall: 0.76
neg precision: 0.756756756757
neg recall: 0.746666666667
```

3. KEYWORDS FOR TRAINING DATA AND HAND LABELED TWEETS AS TEST DATA

a) UNIGRAMS

(1) With Noisy Data

```
train on 1165 instances, test on 300 instances
accuracy: 0.776666666667
Most Informative Features
          great = True          pos : neg    =     5.6 : 1.0
            ass = True          neg : pos    =     5.1 : 1.0
           suck = True          neg : pos    =     5.1 : 1.0
           good = True          pos : neg    =     4.6 : 1.0
            are = True          neg : pos    =     4.5 : 1.0
             in = True          pos : neg    =     4.2 : 1.0
            not = True          neg : pos    =     4.2 : 1.0
            i'm = True          pos : neg    =     4.1 : 1.0
           cool = True          pos : neg    =     3.8 : 1.0
             it = True          pos : neg    =     3.5 : 1.0
pos precision: 0.794326241135
pos recall: 0.746666666667
neg precision: 0.761006289308
neg recall: 0.806666666667
```

(2) Without Noisy Data

```
train on 1165 instances, test on 300 instances
accuracy: 0.773333333333
Most Informative Features
          great = True          pos : neg    =     5.6 : 1.0
            ass = True          neg : pos    =     5.1 : 1.0
           suck = True          neg : pos    =     5.1 : 1.0
           good = True          pos : neg    =     4.6 : 1.0
           cool = True          pos : neg    =     3.8 : 1.0
           like = True          pos : neg    =     3.4 : 1.0
          funny = True          pos : neg    =     3.4 : 1.0
          mambo = True          pos : neg    =     3.4 : 1.0
            big = True          pos : neg    =     3.4 : 1.0
           sana = True          pos : neg    =     2.9 : 1.0
pos precision: 0.797101449275
pos recall: 0.733333333333
neg precision: 0.753086419753
neg recall: 0.813333333333
```

b) **BIGRAMS**

(1) With Noisy Data

```
train on 1165 instances, test on 300 instances
accuracy: 0.69
Most Informative Features
          ('is', 'a') = True          neg : pos    =     2.7 : 1.0
          ('it', 'up') = True         pos : neg    =     2.6 : 1.0
        ('still', 'in') = True        pos : neg    =     1.9 : 1.0
       ('too', 'good') = True            neg : pos    =     1.5 : 1.0
       ('that', 'bad') = True           pos : neg    =     1.1 : 1.0
     ('taking', 'the') = True            pos : neg    =     1.1 : 1.0
        ('just', 'too') = True          pos : neg    =     1.1 : 1.0
        ('2', 'death') = True           pos : neg    =     1.1 : 1.0
     ('my', 'evening') = True              pos : neg    =     1.1 : 1.0
         ('all', 'that') = True         pos : neg    =     1.1 : 1.0
pos precision: 0.813186813187
pos recall: 0.493333333333
neg precision: 0.636363636364
neg recall: 0.886666666667
```

(2) Without Noisy Data

```
train on 1165 instances, test on 300 instances
accuracy: 0.573333333333
Most Informative Features
  ('looking', 'foward') = True          pos : neg    =     1.1 : 1.0
      ('thank', 'god') = None          neg : pos    =     1.0 : 1.0
    ('great', 'stuff') = None          neg : pos    =     1.0 : 1.0
    ('credit', "it's") = None          neg : pos    =     1.0 : 1.0
       ("it's", 'due') = None          neg : pos    =     1.0 : 1.0
       ('got', 'love') = None          neg : pos    =     1.0 : 1.0
    ('fresh', 'death') = None           neg : pos    =     1.0 : 1.0
        ('eat', 'dick') = None         pos : neg    =     1.0 : 1.0
  ('customer', 'care') = None           pos : neg    =     1.0 : 1.0
     ('feeling', 'good') = None         pos : neg    =     1.0 : 1.0
pos precision: 0.958333333333
pos recall: 0.153333333333
neg precision: 0.539855072464
neg recall: 0.993333333333
```

c) **COMBINED UNIGRAMS AND BIGRAMS**

(1) With Noisy Data

```
train on 1165 instances, test on 300 instances
accuracy: 0.803333333333
Most Informative Features
                  great = True           pos : neg    =      5.6 : 1.0
                    ass = True           neg : pos    =      5.1 : 1.0
                   suck = True           neg : pos    =      5.1 : 1.0
                   good = True           pos : neg    =      4.6 : 1.0
                    are = True           neg : pos    =      4.5 : 1.0
                     in = True           pos : neg    =      4.2 : 1.0
                    not = True           neg : pos    =      4.2 : 1.0
                    i'm = True           pos : neg    =      4.1 : 1.0
                   cool = True           pos : neg    =      3.8 : 1.0
                     it = True           pos : neg    =      3.5 : 1.0
pos precision: 0.813793103448
pos recall: 0.786666666667
neg precision: 0.793548387097
neg recall: 0.82
```

(2) Without Noisy Data

```
train on 1165 instances, test on 300 instances
accuracy: 0.773333333333
Most Informative Features
                  great = True           pos : neg    =      5.6 : 1.0
                    ass = True           neg : pos    =      5.1 : 1.0
                   suck = True           neg : pos    =      5.1 : 1.0
                   good = True           pos : neg    =      4.6 : 1.0
                   cool = True           pos : neg    =      3.8 : 1.0
                  funny = True           pos : neg    =      3.4 : 1.0
                  mambo = True           pos : neg    =      3.4 : 1.0
                   like = True           pos : neg    =      3.4 : 1.0
                    big = True           pos : neg    =      3.4 : 1.0
                   sana = True           pos : neg    =      2.9 : 1.0
pos precision: 0.797101449275
pos recall: 0.733333333333
neg precision: 0.753086419753
neg recall: 0.813333333333
```

D. SAMPLE TRAINING AND TEST FEATURES MATRICES

1. Hand Labeled Training Features Matrix – Combined Unigrams and Bigrams

```
: True, 'sepia': True, ('sepia', 'killin'): True}, 'pos'), ({("j'sha", 'lets'):
True, 'lets': True, "j'sha": True}, 'pos'), ({('mix', 'catchy'): True, 'old': Tr
ue, 'skool': True, 'g': True, 'mix': True, ('old', 'skool'): True, 'agai': True,
 ('skool', 'hiphop'): True, 'play': True, ('hiphop', 'mix'): True, ('play', 'aga
i'): True, 'catchy': True, 'hiphop': True, ('g', 'play'): True, ('catchy', 'g'):
 True}, 'pos'), ({('program', 'kindle2'): True, ('idea', '-'): True, 'kindle2':
True, ('awesome', 'idea'): True, '-': True, 'idea': True, ('-', 'continual'): Tr
ue, ('learning', 'program'): True, 'program': True, 'learning': True, 'continual
': True, ('continual', 'learning'): True, 'awesome': True}, 'pos'), ({'40d': Tru
e, 'love': True, ('model', 'getting???'): True, ('love', 'love'): True, 'ooooh':
 True, 'getting???': True, ('getting???', '40d'): True, ('ooooh', 'model'): True
, 'model': True, ('40d', 'love'): True}, 'pos'), ({('dad', 'impressed'): True, '
even': True, ('even', 'dad'): True, 'sol': True, 'sauti': True, 'give': True, ('
gotta', 'give'): True, 'gotta': True, ('sauti', 'sol'): True, ('sol', 'somasoma'
): True, ('video', 'even'): True, 'somasoma': True, ('give', 'sauti'): True, 'vi
deo': True, ('somasoma', 'video'): True, 'dad': True, 'impressed': True}, 'pos')
, ({"india's": True, ("india's", 'election'): True, 'india:': True, 'times': Tru
e, 'election': True, ('times', 'india:'): True, 'wonder': True, ('wonder', "indi
a's"): True, ('india:', 'wonder'): True}, 'pos'), ({('search', 'options'): True,
 ('google', 'using'): True, ('video', 'google'): True, ('good', 'video'): True,
'search': True, 'good': True, 'google': True, 'video': True, ('using', 'search')
: True, 'using': True, 'options': True}, 'pos'), ({'draw': True, 'fashion': True
, 'show': True, 'top': True, 'stanford': True, ('stanford', 'charity'): True, ('
show', 'top'): True, ('fashion', 'show'): True, ('charity', 'fashion'): True, 'c
harity': True, ('top', 'draw'): True}, 'pos'), ({'profile': True, ('official', '
university'): True, ('one', 'popular'): True, ('profile', 'one'): True, 'stanfor
d': True, 'university?s': True, 'university': True, 'official': True, '-': True,
 ('university?s', 'facebook'): True, ('facebook', 'profile'): True, ('pages', '-
'): True, 'facebook': True, 'pages': True, 'popular': True, ('popular', 'officia
l'): True, 'one': True, ('university', 'pages'): True, ('stanford', 'university?
s'): True}, 'pos'), ({('lyx', 'cool'): True, 'lyx': True, 'cool': True}, 'pos'),
 ({'': True, ('danny', 'gokey'): True, 'hero': True, ('gokey', 'home'): True, ('
...danny', ''): True, ('dissapointed', 'sent'): True, '...danny': True, 'rock':
True, 'sooo': True, 'yeah': True, ('sooo', 'dissapointed'): True, ('home', 'stil
l'): True, ('hometown', 'hero'): True, 'home': True, ('rock', '...danny'): True,
 ('yeah', 'milrockee'): True, 'still': True, ('hero', ''): True, ('still', 'rock
'): True, 'gokey': True, 'danny': True, ('', 'yeah'): True, ('sent', 'danny'): T
rue, 'sent': True, 'hometown': True, ('', 'hometown'): True, 'dissapointed': Tru
e, 'milrockee': True}, 'pos'), ({('tones', 'danny'): True, '': True, ('danny', '
gokey'): True, 'danny': True, ('fashion:', 'adam'): True, ('gokey', 'cute'): Tru
e, 'lambert': True, ('lambert', 'tones'): True, ("'american", "idol'"): True, "'
american": True, 'gokey': True, ('adam', 'lambert'): True, 'fashion:': True, '
```

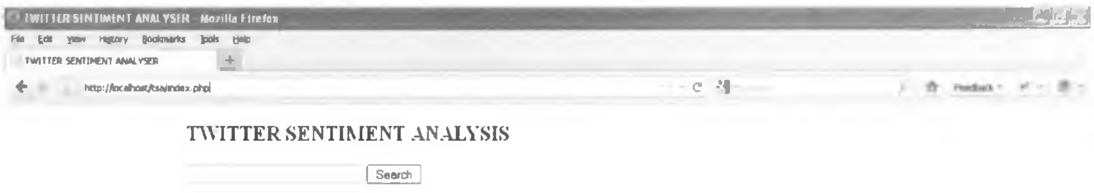
2. Hand Labeled Test Features Matrix – Combined Unigrams and Bigrams

```
'': True, 'people': True, ('coming', 'coz'): True, ('show', 'people'): True, ('p
eople', 'going'): True, 'wit': True, 'request--keep': True, 'coming': True, ('',
 'maaad'): True, 'mins': True, ('em', 'coming'): True, 'maaad': True, 'coz': Tru
e, 'show': True, ('nuts', 'request--keep'): True, ('mins', 'show'): True, 'going
```

```
': True, ('going', 'nuts'): True, ('request--keep', 'em'): True, ('wit', ''): Tr
ue, 'nuts': True, ('coz', 'wit'): True}, 'pos'), ({('paper', 'loving'): True, 'p
aper': True, 'loving': True}, 'pos'), ({'loving': True, ('team', 'selection'): T
rue, 'selection': True, ('loving', 'team'): True, 'team': True}, 'pos'), ({('day
', 'shout'): True, ('til', 'die'): True, 'die': True, ('iv', 'waited'): True, ('
waited', 'day'): True, 'iv': True, 'til': True, ('shout', 'til'): True, 'shout':
 True, 'day': True, 'waited': True}, 'pos'), ({'go': True, ('go', 'gunaz'): True
, ('go', 'go'): True, 'gunaz': True}, 'pos'), ({}, 'pos'), ({'good': True, 'resu
lts': True, ('good', 'results'): True}, 'pos'), ({('result', 'today'): True, '':
 True, ('today', ''): True, 'nice': True, ('nice', 'result'): True, 'today': Tru
e, 'result': True}, 'pos'), ({('theme', 'night'): True, 'town': True, ('town', '
week'): True, 'aiiiiii': True, ("can't", 'wait'): True, 'tuesday': True, ('wait'
, 'next'): True, ('next', 'tuesday'): True, 'week': True, 'next': True, 'theme':
 True, ('night', 'town'): True, 'best': True, 'night': True, ('week', "can't"):
True, ('best', 'theme'): True, ('aiiiiii', 'best'): True, "can't": True, 'wait':
 True}, 'pos'), ({('good', 'great'): True, 'good': True, ('great', 'game'): True
, ('game', 'keep'): True, 'great': True, 'keep': True, 'game': True, 'going': Tr
ue, ('keep', 'going'): True}, 'pos'), ({'ishh': True}, 'pos'), ({('ok', 'give'):
 True, 'ok': True, 'give': True, ('wins', 'one'): True, 'wins': True, 'one': Tru
e, ('give', 'wins'): True}, 'pos'), ({'': True, 'hatrick': True, 'would': True,
'congrats': True, ('would', 'hatrick'): True, ('one', ''): True, ('nice', 'one')
: True, ('', 'congrats'): True, 'one': True, ('asking', 'much.?'): True, 'much.?
': True, ('hatrick', 'asking'): True, 'asking': True, ('congrats', 'brace'): Tru
e, 'brace': True, ('brace', 'would'): True, 'nice': True}, 'pos'), ({('well', 'd
one'): True, ('keep', 'spirit'): True, 'done': True, ('done', 'keep'): True, 'we
ll': True, 'spirit': True, 'keep': True}, 'pos'), ({'congrats': True, ('congrats
', 'fi'): True, 'fi': True, 'di': True, ('fi', 'di'): True, 'win': True, ('di',
'win'): True}, 'pos'), ({('-', 'pun'): True, 'pun': True, ('dinner', '-'): True,
 ('clearly', 'west'): True, 'intended': True, '-': True, ('ham', 'dinner'): True
, 'dinner': True, 'west': True, ('west', 'ham'): True, ('pun', 'intended'): True
, 'clearly': True, 'ham': True}, 'pos'), ({('great', 'keep'): True, 'great': Tru
e, ('playing', 'great'): True, ('well', 'playing'): True, 'well': True, 'playing
': True, 'keep': True}, 'pos'), ({'...on': True, ('second', '...on'): True, 'sec
ond': True, 'vs': True, ('westham', 'vs'): True, ('...on', 'westham'): True, 'we
stham': True}, 'pos'), ({'dance': True, 'looooove': True, 'victory': True, ('vic
tory', 'dance'): True, ('dance', 'looooove'): True}, 'pos'), ({('affleck', 'bar'
): True, ('ths', 'well'): True, 'managed': True, ('ben', 'affleck'): True, 'well
': True, ('bar', 'managed'): True, ('overdue', 'bt'): True, "town'": True, 'bt':
 True, ('well', 'overdue'): True, 'star,direct': True, 'know': True, 'bar': True
, ('know', 'ths'): True, ('managed', 'star,direct'): True, "'the": True, 'ben':
```

E. INSTALLATION INSTRUCTIONS

1. In order to deploy the Twitter Sentiment Analyser, the following prerequisites need to be installed on the target machine.
   - A web server of your choice e.g. Apache Tomcat, IIS etc.
   - At least version 5.3.3 of PHP with curl support.
   - At least version 2.6 of Python installed and initialized into the environment variables (Linux) on Path (Windows).
2. To install the application, please copy the folder labeled "tsa" into your web browser root directory.
3. Ensure that you have a direct connection to the internet from the target machine. If you do, skip to step number 6 otherwise proceed to step number 4.
4. If you do not have a direct connection to the internet, you are probably behind a proxy. Navigate into "tsa/includes" and find the files "class.twittersearch.php". Open this file and go to line number 309. Uncomment this line and specify your proxy IP address and port.
5. If a proxy username and password is required, uncomment line 310 and add the necessary proxy credentials.
6. Ensure your web server is started and point your browser to this address http://[nameofserver]/tsa/index.php where [nameofserver] could be the hostname, or IP address of the target machine. E.g. http://localhost/tsa/index.php .If you receive the page displayed below the installation has successfully completed. You can now proceed to enter the queries you wish to search in Twitter, for sentiment analysis results.


TWITTER SENTIMENT ANALYSER - Mozilla Firefox
File Edit View History Bookmarks Tools Help
TWITTER SENTIMENT ANALYSER
http://localhost/tsa/index.php
TWITTER SENTIMENT ANALYSIS
Search