SENTIMENT ANALYSIS OF TWEETS ABOUT BIG BILLION DAYS

B.Tech. Minor Project Report

BY

N RITVIKA REDDY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

RAIPUR- CG (INDIA)

DECEMBER, 2015

SENTIMENT ANALYSIS OF TWEETS ABOUT BIG BILLION DAYS

A Minor Project Report

submitted in partial fulfillment of the

requirements for the award of the degree

of

Bachelor of Technology

in

COMPUTER SC. & ENGINEERING

BY

N RITVIKA REDDY

DEPARTMENT OF COMPUTER SC. & ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

RAIPUR , CG (INDIA)

DECEMBER, 2015

CERTIFICATE

I hereby certify that the work which is being presented in the B.Tech. Major (or Minor) Project Report entitled **"Sentiment Analysis of Tweets about Big Billion Days",** in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Computer Sc. & Engineering** and submitted to the Department of Computer Sc. & Engineering of National Institute of Technology Raipur is an authentic record of my own work carried out during a period from July 2015 to December 2015 under the supervision of **Mr. Dilip Singh Sisodia, Assistant Professor, CSE Department**.

The matter presented in this thesis has not been submitted by me for the award of any other degree elsewhere.

Signature of Candidate

N RITVIKA REDDY

R.No. 12115045

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Signature of Supervisor(s)

Mr. Dilip Singh Sisodia

Assistant Professor

Head

Computer Sc. & Engineering Department

National Institute of Technology Raipur CG

ACKNOWLEDGEMENT

I am deeply thankful to my project supervisor, Mr.Dilip Singh Sisodia for helping me throughout the course in accomplishing this project. His enthusiastic guidance and support enabled me in achieving the objectives of the project.

ABSTRACT

Customer opinions play a very crucial role in daily life. Whenever someone has to take a decision, they tend to consider opinions of others. Analysing the public sentiment is important for many applications such as firms trying to find out the response of their products in the market, predicting political elections and predicting socioeconomic phenomena like stock exchange. Since there is a good amount of social data present on the Web automatically therefore, its very important to employ methods that automatically classify them. Sentiment Classification sometimes called as Opinion Mining is defined as mining and analyzing of reviews, views, emotions and opinions automatically from text, big data and speech by means of various methods. This project addresses sentiment analysis in twitter; that is classifying tweets according to the sentiment expressed in them: positive, negative or neutral.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER - 1
INTRODUCTION

1.1 Introduction To Sentiment Analysis:

In the past decade, new forms of communication, such as microblogging and text messaging have emerged and become ubiquitous. Twitter [34] is one such online platform which allows users to write short status updates of maximum length 140 characters. These short messages are used to share opinions and sentiments that people have about what is going on in the world around them.

In this chapter, we give a brief introduction about what is Opinion Mining and how it can be performed. Sentiment classification or Opinion Mining involves building a system to make use of reviews posted by the users and opinions that are expressed in blogs and forums as comments and reviews and sometimes may be as tweets about the product.

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Sentiment analysis is widely applied to reviews and social media for a variety of applications, ranging from marketing to customer service. Sentiment analysis includes in building a framework to gather and look at opinions about the item made in blog entries, remarks, audits or tweets.

To be specific, term Sentiment is exceptionally wide and it constitutes feelings, opinions, dispositions, particular encounters, and so forth. In this theory, we speak just about the opinions communicated in writings which are composed in texts which are written in human readable natural language, in social media.

In this project, we focus on tweets posted by the public with regard to “The Big Billion Days” sale conducted by Flipkart [35]. We have chosen to work with Twitter since we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for Twitter, as compared to traditional blogging sites. Moreover the response on Twitter is more prompt and also more general (since the number of users who tweet is substantially more than those who write web blogs on a daily basis). Sentiment analysis of public is highly critical in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm.

Tweets and texts are short: a sentence or a headline rather than a document. The language used is very informal, with creative spelling and punctuation,misspellings, slang, new words, URLs, and genre-specific terminology and abbreviations, such as, RT for "re-tweet" and # hashtags, which are a type of tagging for Twitter messages. How to handle such challenges so as to automatically mine and understand the opinions and sentiments that people are communicating has been the subject of many researchers.

This project implements the techniques of “Natural Language Processing” in extracting significant patterns and features from the large data set of tweets and “Machine Learning” techniques for accurately classifying individual unlabelled data samples (tweets) according to whichever pattern model best describes them.

The features that can be used for modeling patterns and classification can be divided into two main groups: formal language based and informal blogging based. Language based features are those that deal with formal linguistics and include prior sentiment polarity of individual words and phrases, and parts of speech tagging of the sentence. Prior sentiment polarity means that some words and phrases have a natural innate tendency for expressing particular and specific sentiments in general. For example the word “excellent” has a strong positive connotation while the word “evil” possesses a strong negative connotation. So whenever a word with positive connotation is used in a sentence, chances are that the entire sentence would be expressing a positive sentiment. Twitter based features are more informal and relate with how people express themselves on online social platforms and compress their sentiments in the limited space of 140 characters offered by twitter. They include twitter hashtags, retweets, word capitalization, word lengthening, question marks, presence of url in tweets, exclamation marks, internet emoticons and internet shorthand/slangs.

Classification techniques can also be divided into a two categories: Supervised vs. unsupervised and non-adaptive vs. adaptive/reinforcement techniques. Supervised approach is when we have pre-labeled data samples available and we use them to train our classifier. Training the classifier means to use the pre-labeled to extract features that best model the patterns and differences between each of the individual classes, and then classifying an unlabeled data sample according to whichever pattern best describes it.

Unsupervised classification is when we do not have any labeled data for training. In addition to this adaptive classification techniques deal with feedback from the environment.

There are two main methods of sentiment analysis: lexicon-based methods [17] and machine learning methods. Current works preferably focuses on supervised learning approach to sentiment classification. We have three existing machine learning methods namely Naive Bayes, Maximum Entropy classification, and Support Vector Machine (SVM) to classify texts as positive or negative. We infer that the standard machine learning methods perform far better than human produced approaches.

CHAPTER - 2
LITERARY REVIEW

2.1 Limitations Of Previous Works:

Sentiment analysis is an area of research that investigates people's opinions towards different matters: products, events, organisations [1]. Decent amount of related prior work has been done on sentiment analysis of user reviews [2], documents, web blogs/articles and general phrase level sentiment analysis [3]. These differ from twitter mainly because of the limit of 140 characters per tweet which forces the user to express opinion compressed in very short text. The best results reached in sentiment classification use supervised learning techniques such as Naive Bayes and Support Vector Machines, but the manual labelling required for the supervised approach is very expensive. Some work has been done on unsupervised [4] [5] and semi-supervised [6] [7] approaches, and there is a lot of room of improvement. Various researchers testing new features and classification techniques often just compare their results to base-line performance. There is a need of proper and formal comparisons between these results arrived through different features and classification techniques in order to select the best features and most efficient classification techniques for particular applications.

2.2 Related Works:

The bag-of-words model is one of the most widely used feature model for almost all text classification tasks due to its simplicity coupled with good performance. The model represents the text to be classified as a bag or collection of individual words with no link or dependence of one word with the other, i.e. it completely disregards grammar and order of words within the text. This model is also very popular in sentiment analysis and has been used by various researchers. The simplest way to incorporate this model in our classifier is by using unigrams as features. Generally speaking n-grams is a contiguous sequence of "n" words in our text, which is completely independent of any other words or grams in the text. So unigrams is just a collection of individual words in the text to be classified, and we assume that the probability of occurrence of one word will not be affected by the presence or absence of any other word in the text. This is a very simplifying assumption but it has been shown to provide rather good performance [8] [9]. One simple way to use unigrams as features is to assign them with a certain prior polarity, and take the average of the overall polarity of the text, where the overall polarity of the text could simply be calculated by summing the prior polarities of individual unigrams.

There are three ways of using prior polarity of words as features. The simpler un-supervised approach is to use publicly available online lexicons/dictionaries which map a word to its prior polarity. The Multi-Perspective-Question-Answering (MPQA) is an online resource with such a subjectivity lexicon which maps a total of 4,850 words according to whether they are “positive” or “negative” and whether they have “strong” or “weak” subjectivity [36]. The SentiWordNet 3.0 is another such resource which gives probability of each word belonging to positive, negative and neutral classes [10]. The second approach is to construct a custom prior polarity dictionary from our training data according to the occurrence of each word in each particular class. For example if a certain word is occurring more often in the positive labelled phrases in our training dataset (as compared to other classes) then we can calculate the probability of that word belonging to positive class to be higher than the probability of occurring in any other class. This approach has been shown to give better performance, since the prior polarity of words is more suited and fitted to a particular type of text and is not very general like in the former approach. However, the latter is a supervised approach because the training data has to be labelled in the appropriate classes before it is possible to calculate the relative occurrence of a word in each of the class. Kouloumpis et al. noted a decrease in performance by using the lexicon word features along with custom n-gram word features constructed from the training data, as opposed to when the n-grams were used alone [8]. The third approach is a middle ground between the above two approaches. In this approach we construct our own polarity lexicon but not necessarily from our training data, so we don’t need to have labelled training data. One way of doing this as proposed by Turney et al. is to calculate the prior semantic orientation (polarity) of a word or phrase by calculating it’s mutual information with the word “excellent” and subtracting the result with the mutual information of that word or phrase with the word “poor” [4].

Many of the researchers in this field have used already constructed publicly available lexicons of sentiment bearing words [3] [8] [11] while many others have also explored building their own prior polarity lexicons [4] [6] [7].

The basic problem with the approach of prior polarity approach has been identified by Wilson et al. who distinguish between prior polarity and contextual polarity [3]. They say that the prior polarity of a word may in fact be different from the way the word has been used in the particular context. The paper explores some other features which include grammar and syntactical relationships between words to make their classifier better at judging the contextual polarity of the phrase.

The task of twitter sentiment analysis can be most closely related to phrase-level sentiment analysis. A seminal paper on phrase level sentiment analysis was presented in 2005 by Wilson et al. [3] which identified a new approach to the problem by first classifying phrases according to subjectivity (polar) and objectivity (neutral) and then further classifying the subjective-classified phrases as either positive or negative. The paper noticed that many of the objective phrases used prior sentiment bearing words in them, which led to poor classification of especially objective phrases. It claims that if we use a simple classifier which assumes that the contextual polarity of the word is merely equal to its prior polarity gives a result of about 48%. The novel classification process proposed by this paper along with the list of ingenious features which include information about contextual polarity resulted in significant improvement in performance (in terms of accuracy) of the classification process.

One way of alleviating the condition of independence and including partial context in our word models is to use bigrams and trigrams as well besides unigrams. Bigrams are collection of two contiguous words in a text, and similarly trigrams are collection of three contiguous words. So we could calculate the prior polarity of the bigram / trigram - or the prior probability of that bigram / trigram belonging to a certain class – instead of prior polarity of individual words. Many researchers have experimented with them with the general conclusion that if we have to use one of them alone unigrams perform the best, while unigrams along with bigrams may give better results with certain classifiers [6] [9]. However trigrams usually result in poor performance as reported by Pak et al. [6]. The reduction in performance by using trigrams is because there is a compromise between capturing more intricate patterns and word coverage as one goes to higher-numbered grams. Besides from this some researchers have tried to incorporate negation into the unigram word models. Pang et al. and Pak et al. used a model in which the prior polarity of the word was reversed if there was a negation (like "not", "no", "don't", etc.) next to that word [2] [6]. In this way some contextual information is included in the word models.

Besides from these much work has been done in exploring a class of features pertinent only to micro blogging domain. Presence of URL and number of capitalized words/alphabets in a tweet have been explored by Koulompis et al. [8] and Barbosa et al. [7]. Koulmpis also reports positive results for using emoticons and internet slang words as features. Brody et al. does study on word lengthening as a sign of subjectivity in a tweet [5]. The paper reports positive results for their study that the more number of cases a word has of lengthening, the more chance there of that word being a strong indication of subjectivity.

The most commonly used classification techniques are the Naive Bayes Classifier and State Vector Machines. Some researchers like Barbosa et al. publish better results for SVMs [7] while others like Pak et al. support Naive Bayes [6].

It has been observed that having a larger training sample pays off to a certain degree, after which the accuracy of the classifier stays almost constant even if we keep adding more labelled tweets in the training data [7]. Barbosa et al. used tweets labelled by internet resources [37], instead of labelling them by hand, for training the classifier. Although there is loss of accuracy of the labelled samples in doing so (which is modelled as increase in noise) but it has been observed that if the accuracy of training labels is greater than 50%, the more the labels, the higher the accuracy of the resulting classifier. So in this way if there are an extremely large number of tweets, the fact that our labels are noisy and inaccurate can be compensated for [7]. On the other hand Pak et al. and Go et al. [9] use presence of positive or negative emoticons to assign labels to the tweets [6]. Like in the above case they used large number of tweets to reduce effect of noise in their training data.

Some of the earliest work in this field classified text only as positive or negative, assuming that all the data provided is subjective [2] [9]. While this is a good assumption for something like movie reviews but when analyzing tweets and blogs there is a lot of objective text we have to consider, so incorporating neutral class into the classification process is now becoming a norm. Some of the work which has included neutral class into their classification process includes [3] [6] [7] [8].

There has also been very recent research of classifying tweets according to the mood expressed in them, which goes one step further. Bollen et al. explores this area and develops a technique to classify tweets into six distinct moods: tension, depression, anger, vigour, fatigue and confusion [12]. They use an extended version of Profile of Mood States (POMS): a widely accepted psychometric instrument. They generate a word dictionary and assign them weights corresponding to each of the six mood states, and then they represented each tweet as a vector corresponding to these six dimensions. However not much detail has been provided into how they built their customized lexicon and what technique did they use for classification.

Sentiment analysis has not only been employed for analysing movie and product reviews but also for predicting the results of popular political elections and polls is also an emerging application to sentiment analysis. One such study was conducted by Tumasjan et al. in Germany for predicting the outcome of federal elections in which concluded that Twitter is a good reflection of offline sentiment [13]. Another study which was conducted by Prasetyo et al. in Indonesia during the 2014 Presidential Elections concluded that Twitter-based election prediction, in a developing country such as Indonesia, offers competitive prediction accuracy [14]. Location based Sentiment Analysis was conducted by Almatrafi et al. regarding the 2014 Indian General Elections which concluded that sentiments change from one location to the other and also that certain social events can trigger a sharp rise in the sentiments of the public [15].

CHAPTER - 3
PROCESS FLOW

3.1 Steps Involved:

The steps involved in the entire process of classifying the sentiment of the tweets are depicted in Figure 3.1.

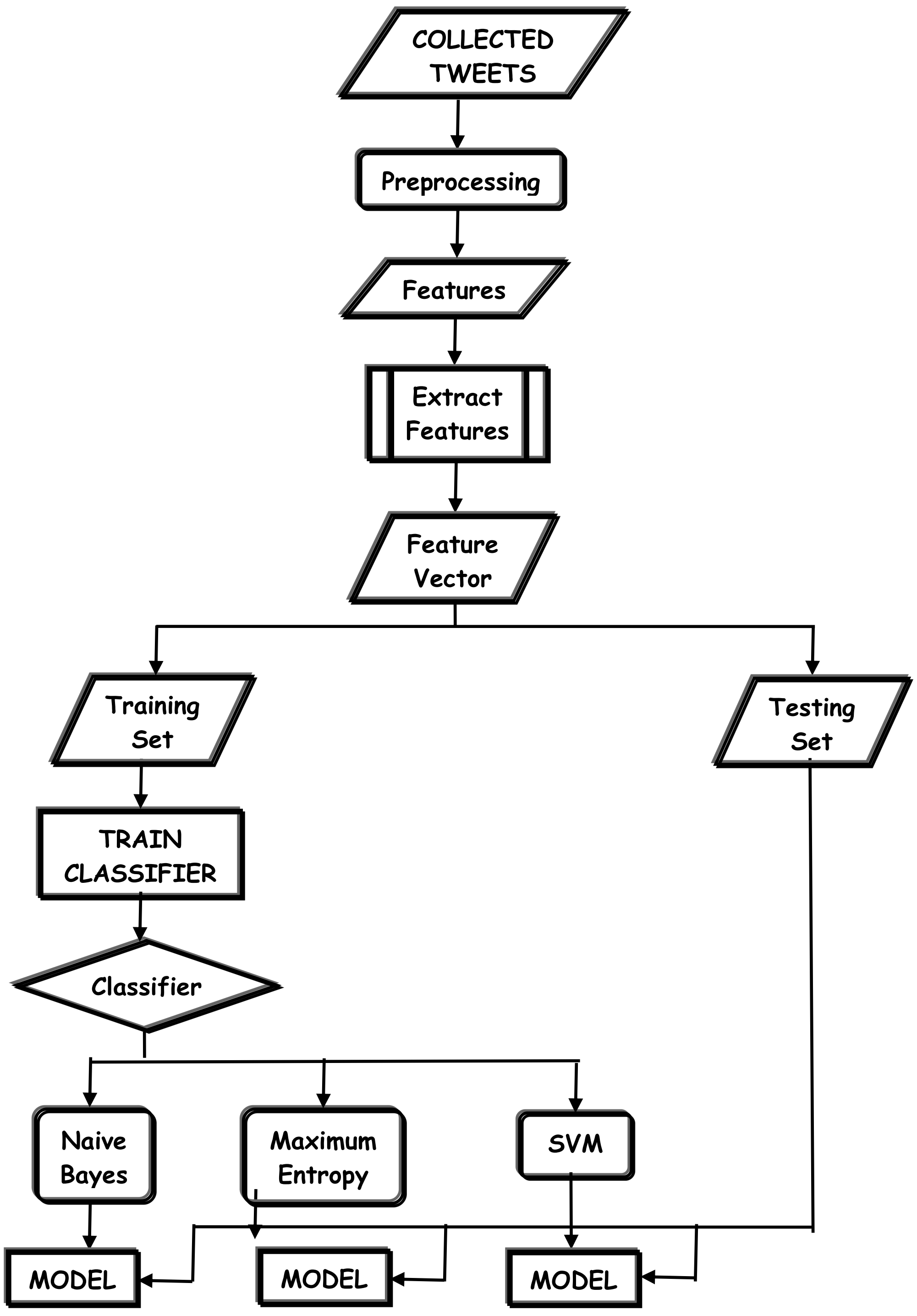


Figure 3.1 : Process Flow Chart

We start our project by collecting the data required. The entire procedure of gathering the data is elaborated in 4.1 and 4.2 of Chapter 4. We collect only those tweets which contain either the ‘#thebigbilliondays’ or ‘#bigbilliondays’ hashtags.

The next step is to parse the JSON encoded tweet into a format which is useful to us by filtering attributes of the tweet through the keys given. 4.2 of Chapter 4 states the manner in which we parse the tweet. All the tweets that are collected are divided into 2 sets after. One set ,called the training-set, is used for training the classifiers. The tweets of the training set are manually annotated and all the irrelevant tweets are eliminated. The other set is used as a test-set for the classifiers The training-set is a balanced set of 486 tweets and the testing-set contains of 1068 tweets in all.

After parsing the tweets, we have to preprocess the text of the tweets. The steps followed in preprocessing are mentioned in 4.3 of Chapter 4.

On preprocessing the tweets, we extract the features of the tweet into a feature vector. Each and every tweet is associated with a feature vector. Then, we create a feature list from all the feature vectors of the tweets belonging to the training-set. The classifiers are trained and then the test-set is passed as input to the classifiers for classification.

Polarity	**Training Data**
Positive	162
Negative	162
Neutral	162

Table 3.1 : Composition of Training Data

We use the Naive Bayes classifier, the Maximum Entropy classifier and the SVM classifier. Chapter 5 contains the principles and algorithms of these three classifiers.We compare the performances of the classifiers based on their respective precision, recall and F-measure. The results are given in Chapter 6.

CHAPTER - 4

DATA COLLECTION AND PREPROCESSING

4.1 Collecting Data:

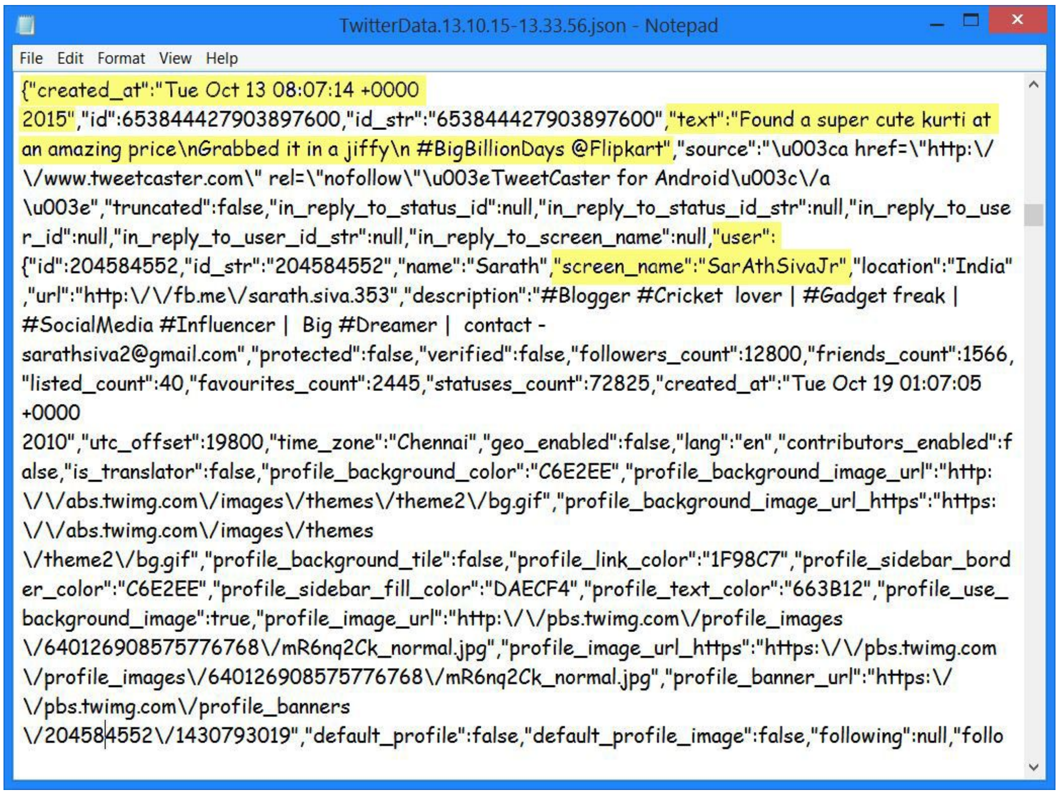
As this project aims to classify the sentiment of the public regarding “The Big Billion Days” sale organised by Flipkart, the tweets used were collected using the Tweepy API over a period of 5 days from 13th Oct,2015 to 17th Oct,2015 [38]. Tweepy is an open-source API which is hosted on GitHub [39] and enables Python [40] to communicate with Twitter platform and use its API.

4.2 Using the Tweepy API:

Tweepy supports accessing Twitter via Basic Authentication and the newer method, OAuth. With Tweepy, it's possible to get any object and use any method that the official Twitter API [41] offers. Main Model classes in the Twitter API are Tweets, Users, Entities and Places. Access to each returns a JSON-formatted response and traversing through information is very easy in Python. One of the main usage cases of Tweepy is monitoring for tweets and doing actions when some event happens. Key component of that is the StreamListener object, which monitors tweets in real time and catches them.

Once we connect to the Twitter API, the data we get back will be encoded in JSON, (or JavaScript Object Notation). JSON is a way to encode complicated information in a platform-independent way. It is a rather simplistic and elegant way to encode complex data structures. Figure 4.1 shows how a tweet encoded in JSON looks like.

The JSON format is specified in terms of key/value pairs. There are a list of keys present in the JSON format of the tweets like: user, favorited, contributors, entities, text, created_at, user, truncated, retweeted, in_reply_to_status_id, coordinates, id, source, in_reply_to_status_id_str, in_reply_to_screen_name, id_str, place, retweet_count, geo, in_reply_to_user_id_str, and in_reply_to_user_id. The user key another list of keys, with per-user information for each tweet: profile_use_background_image, geo_enabled, description, followers_count, statuses_count, friends_count, location, name, screen_name.


TwitterData.13.10.15-13.33.56.json - Notepad
File Edit Format View Help

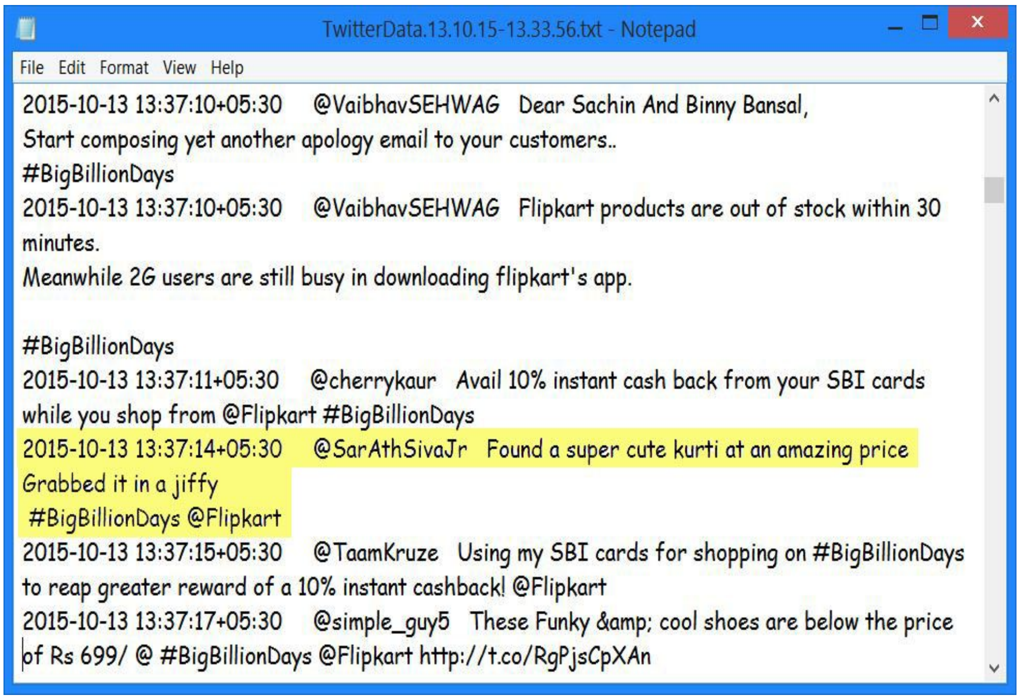
```
{"created_at":"Tue Oct 13 08:07:14 +0000
2015","id":653844427903897600,"id_str":"653844427903897600","text":"Found a super cute kurti at
an amazing price\nGrabbed it in a jiffy\n #BigBillionDays @Flipkart","source":"\u003ca href=\"http:\/
\/www.tweetcaster.com\" rel=\"nofollow\"\u003eTweetCaster for Android\u003c\/a
\u003e","truncated":false,"in_reply_to_status_id":null,"in_reply_to_status_id_str":null,"in_reply_to_use
r_id":null,"in_reply_to_user_id_str":null,"in_reply_to_screen_name":null,"user":
{"id":204584552,"id_str":"204584552","name":"Sarath","screen_name":"SarAthSivaJr","location":"India"
,"url":"http:\/\/fb.me\/sarath.siva.353","description":"#Blogger #Cricket  lover | #Gadget freak |
#SocialMedia #Influencer |  Big #Dreamer |  contact -
sarathsiva2@gmail.com","protected":false,"verified":false,"followers_count":12800,"friends_count":1566,
"listed_count":40,"favourites_count":2445,"statuses_count":72825,"created_at":"Tue Oct 19 01:07:05
+0000
2010","utc_offset":19800,"time_zone":"Chennai","geo_enabled":false,"lang":"en","contributors_enabled":f
alse,"is_translator":false,"profile_background_color":"C6E2EE","profile_background_image_url":"http:
\/\/abs.twimg.com\/images\/themes\/theme2\/bg.gif","profile_background_image_url_https":"https:
\/\/abs.twimg.com\/images\/themes
\/theme2\/bg.gif","profile_background_tile":false,"profile_link_color":"1F98C7","profile_sidebar_bord
er_color":"C6E2EE","profile_sidebar_fill_color":"DAECF4","profile_text_color":"663B12","profile_use_
background_image":true,"profile_image_url":"http:\/\/pbs.twimg.com\/profile_images
\/640126908575776768\/mR6nq2Ck_normal.jpg","profile_image_url_https":"https:\/\/pbs.twimg.com
\/profile_images\/640126908575776768\/mR6nq2Ck_normal.jpg","profile_banner_url":"https:\/
\/pbs.twimg.com\/profile_banners
\/204584552\/1430793019","default_profile":false,"default_profile_image":false,"following":null,"follo
```

Figure 4.1 : JSON Encoded Tweet

From this data, we filter out only those attributes of the tweet which are useful for us. They are :

i. [created_at]
ii. [text]
iii. [user] [screen_name]

For every tweet that we have collected , we pull out these keys. On collecting all these tweets we save them into a text file. The filtered data is shown in Figure 4.2.


TwitterData.13.10.15-13.33.56.txt - Notepad
File Edit Format View Help

```
2015-10-13 13:37:10+05:30	@VaibhavSEHWAG	Dear Sachin And Binny Bansal,
Start composing yet another apology email to your customers..
#BigBillionDays
2015-10-13 13:37:10+05:30	@VaibhavSEHWAG	Flipkart products are out of stock within 30 minutes.
Meanwhile 2G users are still busy in downloading flipkart's app.

#BigBillionDays
2015-10-13 13:37:11+05:30	@cherrykaur	Avail 10% instant cash back from your SBI cards while you shop from @Flipkart #BigBillionDays
2015-10-13 13:37:14+05:30	@SarAthSivaJr	Found a super cute kurti at an amazing price
Grabbed it in a jiffy
 #BigBillionDays @Flipkart
2015-10-13 13:37:15+05:30	@TaamKruze	Using my SBI cards for shopping on #BigBillionDays to reap greater reward of a 10% instant cashback! @Flipkart
2015-10-13 13:37:17+05:30	@simple_guy5	These Funky &amp; cool shoes are below the price of Rs 699/ @ #BigBillionDays @Flipkart http://t.co/RgPjsCpXAn
```

Figure 4.2 : Parsed JSON Tweet

4.3 Pre-processing Of Data:

Before applying any of the sentiment extraction methods, it is a common practice to perform data pre-processing. Data pre-processing is done to eliminate the incomplete, noisy and inconsistent data. Data must be pre-processed in order to perform any data mining functionality. The methods involved in are shown in Figure 4.3 :

4.3.1 Removal of Stop-Words:

Stop words are words which carry a connecting function in the sentence, such as prepositions, articles, etc. [16]. Some of the most common words like “the”,“at”,“is”, etc. can be removed from the tweets since they do not contribute to the sentiment of the tweet.

4.3.2 Removing Special Characters:

Special characters like.,[]{}()/’ should be removed in order to remove discrepancies during the assignment of polarity.

4.3.3 Removal of Retweets:

Retweeting is the process of copying another user's tweet and posting to another account. This usually happens if a user likes another user's tweet. Retweets are commonly abbreviated with “ RT ”.

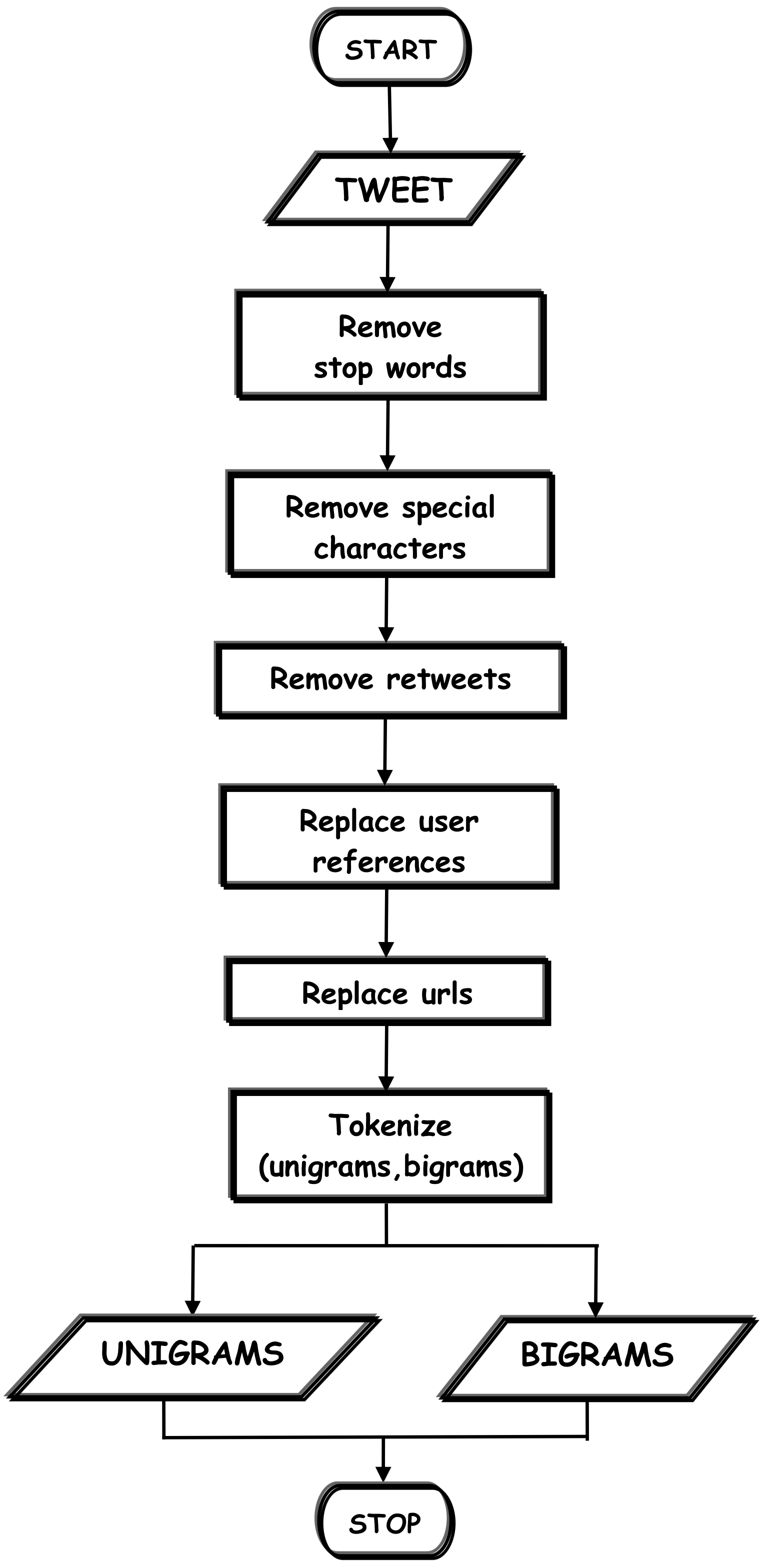


Figure 4.3 : Flow Chart for Preprocessing Tweet Text

4.3.4 Replacing user references:

In tweets, many users tend to add the names of other users when directing tweets at them. Since these references do not add to the sentiment of the tweet, we replace these names with a constant word like “USER”. Tweets contain user-names that are usually prefixed by an “@” symbol.

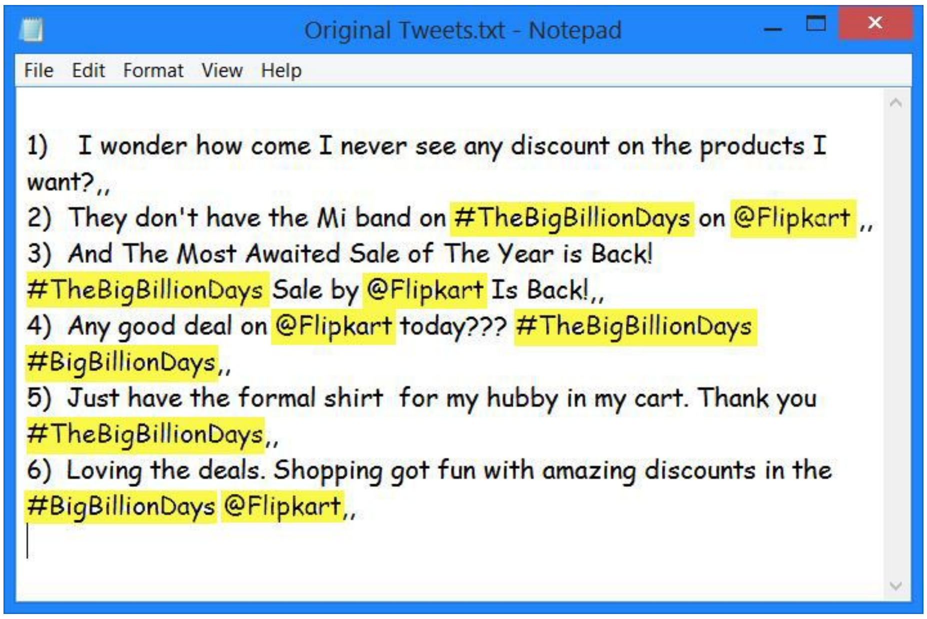
4.3.5 Replacing urls:

Many tweets like normal text might contain links of websites which are do not affect the overall sentiment of the tweet. Hence , we can replace these links by using a constant like “URL”. Urls can be identified as strings beginning with “http” or “www”.

4.3.6 Tokenisation into N-grams:

Tokenisation is a process of creating a bag-of-words from the text. The incoming string gets broken into comprising words and other elements, for example URL links. The common separator for identifying individual words is white-space, however other symbols can also be used. Tokenisation of social-media data is considerably more difficult than tokenisation of the general text since it contains numerous emoticons, URL links, abbreviations that cannot be easily separated as whole entities.

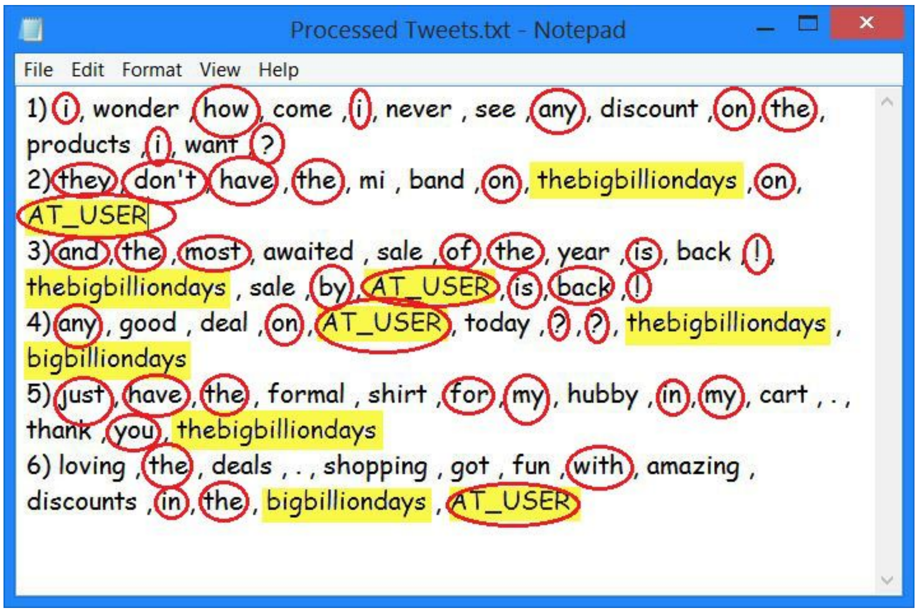
Figure 4.4 shows a sample of the collected tweets after parsing. The highlighted words in the figure correspond to the user references and hashtags.



1) I wonder how come I never see any discount on the products I want?,,
2) They don't have the Mi band on #TheBigBillionDays on @Flipkart ,,
3) And The Most Awaited Sale of The Year is Back! #TheBigBillionDays Sale by @Flipkart Is Back!,,
4) Any good deal on @Flipkart today??? #TheBigBillionDays #BigBillionDays,,
5) Just have the formal shirt for my hubby in my cart. Thank you #TheBigBillionDays,,
6) Loving the deals. Shopping got fun with amazing discounts in the #BigBillionDays @Flipkart,,

Figure 4.2 : Collected Tweets

In Figure 4.5, we can see that the highlighted words of Figure 4.4 have now been replaced. The circled words in this figure correspond to the stop-words.

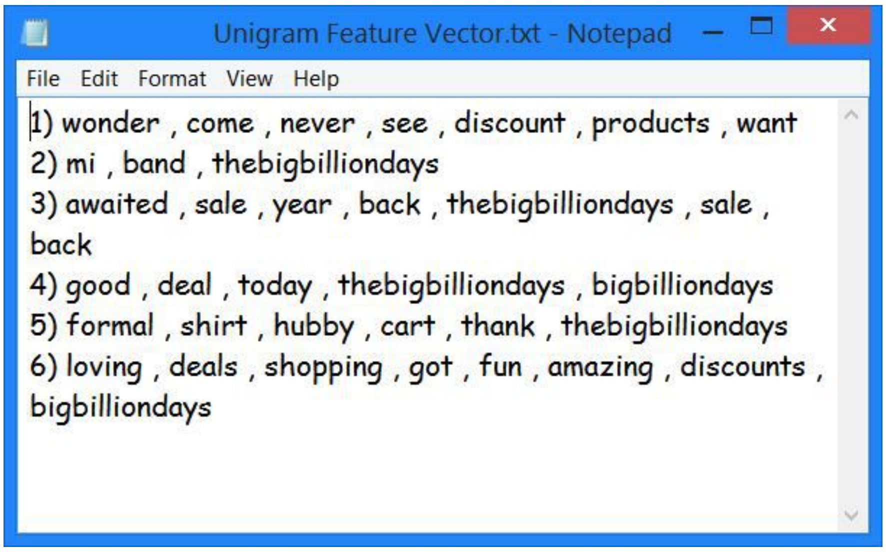

Processed Tweets.txt - Notepad

File Edit Format View Help

```
1) i , wonder , how , come , i , never , see , any , discount , on , the ,
products , i , want , ?
2) they , don't , have , the , mi , band , on , thebigbilliondays , on ,
AT_USER
3) and , the , most , awaited , sale , of , the , year , is , back , ! ,
thebigbilliondays , sale , by , AT_USER , is , back , !
4) any , good , deal , on , AT_USER , today , ? , ? , thebigbilliondays ,
bigbilliondays
5) just , have , the , formal , shirt , for , my , hubby , in , my , cart , . ,
thank , you , thebigbilliondays
6) loving , the , deals , . , shopping , got , fun , with , amazing ,
discounts , in , the , bigbilliondays , AT_USER
```

Figure 4.5 : User References and Hashtags Filtering

In Figure 4.6 , we can see that the stop-words circled in Figure 4.5 have been eliminated and we are left with Unigram Feature Vector.

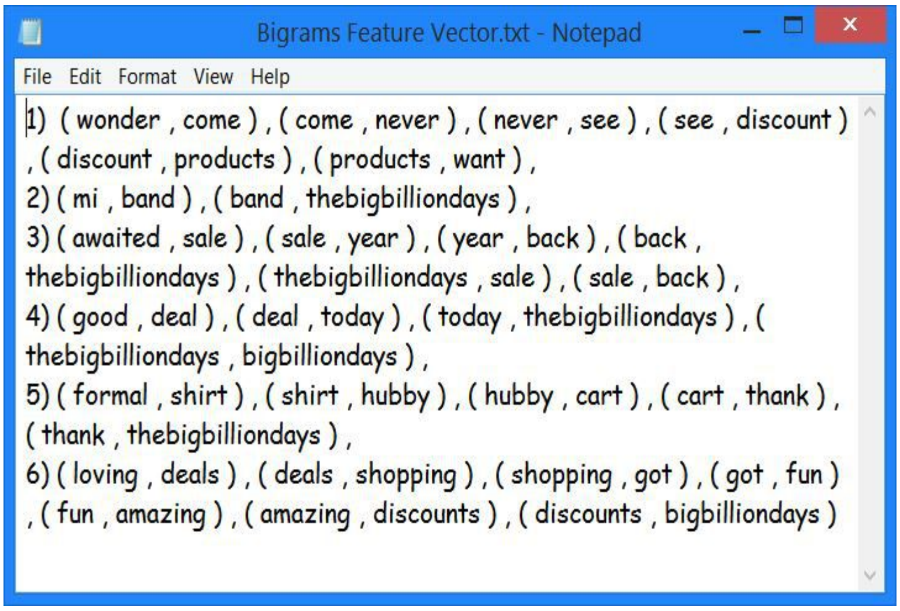

Unigram Feature Vector.txt - Notepad

File Edit Format View Help

```
1) wonder , come , never , see , discount , products , want
2) mi , band , thebigbilliondays
3) awaited , sale , year , back , thebigbilliondays , sale ,
back
4) good , deal , today , thebigbilliondays , bigbilliondays
5) formal , shirt , hubby , cart , thank , thebigbilliondays
6) loving , deals , shopping , got , fun , amazing , discounts ,
bigbilliondays
```

Figure 4.6 : Stop Word Removal and Tokenization into Unigrams

Similarly Figure 4.7, we can see that the stop-words circled in Figure 4.5 have been eliminated and we are left with Bigram Feature Vector.


Bigrams Feature Vector.txt - Notepad

File Edit Format View Help

```
1) ( wonder , come ) , ( come , never ) , ( never , see ) , ( see , discount )
, ( discount , products ) , ( products , want ) ,
2) ( mi , band ) , ( band , thebigbilliondays ) ,
3) ( awaited , sale ) , ( sale , year ) , ( year , back ) , ( back ,
thebigbilliondays ) , ( thebigbilliondays , sale ) , ( sale , back ) ,
4) ( good , deal ) , ( deal , today ) , ( today , thebigbilliondays ) , (
thebigbilliondays , bigbilliondays ) ,
5) ( formal , shirt ) , ( shirt , hubby ) , ( hubby , cart ) , ( cart , thank ) ,
( thank , thebigbilliondays ) ,
6) ( loving , deals ) , ( deals , shopping ) , ( shopping , got ) , ( got , fun )
, ( fun , amazing ) , ( amazing , discounts ) , ( discounts , bigbilliondays )
```

Figure 4.7 : Stop Word Removal and Tokenization into Bigrams

CHAPTER - 5
METHODS USED

5. METHODS USED:

In this project, I have employed the Naive Bayes Classifier and the MaxEnt Classifier present in the Natural Language Toolkit (NLTK) package for Python [42].

5.1 Naive Bayes Classifier:

The Bayesian Classification represents a supervised learning method as well as a statistical method for classification. Assumes an underlying probabilistic model and it allows us to capture uncertainty about the model in a principled way by determining probabilities of the outcomes. This classification is named after Thomas Bayes (1702-1761), who proposed the Bayes Theorem.

Bayesian reasoning is applied to decision making and inferential statistics that deals with probability inference. It is used the knowledge of prior events to predict future events. So, given the training data which contains the number of positive and negative labelled tweets, the Naive Bayes Classifier can predict the polarity of any tweet given as a test case.

The Naive Bayesian classifier is based on Bayes' theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

Bayes theorem provides a way of calculating the posterior probability, $P(c \mid x)$, from $P(c)$, $P(x)$, and $P(x \mid c)$. Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)} \qquad [5.1a]$$

$P(c \mid x)$ is the posterior probability of class (target) given predictor (attribute).
$P(c)$ is the prior probability of class.
$P(x \mid c)$ is the likelihood which is the probability of predictor given class.
$P(x)$ is the prior probability of predictor.

Naive Bayes Classification is based on the Bayesian theorem equation given in [5.1a]. This can be simplified by assuming that features are independent when given the class [18]. This is depicted in [5.1b]. Despite this assumption, the classifier is remarkably successful in practice [19][20][21].

$$P(x \mid c) = \prod_{i=1}^{n} P(x_i \mid c) \qquad [5.1b]$$

where $x = \{x_1, \ldots x_n\}$ is a feature vector.

The pseudo code for the Naive Bayes Classifier according to [32] is as follows :

```
function train(i)    {
   Instances++
   if (++N[$Klass]==1) Klasses++
   for(i=1;i<=Attr;i++)
     if (i != Klass)
       if ($i !~ /\?/)
          symbol(i,$i,$Klass)
}

function symbol(col,value,klass)    {
   Count[klass,col,value]++;
}
```

When testing, it finds the likelihood of each hypothetical class and returns the one that is most likely.

```
function likelihood(l,klass,i,inc,temp,prior,what,like) {
 like = -10000000000;      # smaller than any log
 for(klass in N) {
   prior=N[klass] / Instances;
       temp= prior
       for(i=1;i<=Attr;i++) {
          if (i != Klass)
             if ( $i !~ /\?/ )
                 temp *= Count[klass,i,$i] / N[klass]
       }
       l[klass]= temp
       if ( temp >= like ) {like = temp; what=klass}
   }
   return what
}
```

It is particularly suited when the dimensionality of the inputs is high. Parameter estimation for naive Bayes models uses the method of maximum likelihood. In spite over-simplified assumptions, it often performs better in many complex real-world situations. In machine learning, considerable theoretical and experimental evidence has been developed that a training procedure based on the Naive Bayes assumptions can yield an optimal classifier in a variety of situations where the assumptions are wildly violated [21]. Recent comparisons of machine learning methods for text categorization have been somewhat less favorable to naive Bayes models while still showing them to achieve respectable effectiveness [22][23].

5.2 Maximum Entropy Classifier :

The principle of maximum entropy states that, subject to precisely stated prior data the probability distribution which best represents the current state of knowledge is the one with largest entropy.

Maximum entropy (MaxEnt) models have been used in many spoken language tasks. The training of a MaxEnt model often involves an iterative procedure that starts from an initial parameterization and gradually updates it towards the optimum. Due to the convexity of its objective function (hence a global optimum on a training set), little attention has been paid to model initialization in MaxEnt training [24].

The Max Entropy classifier is a probabilistic classifier which belongs to the class of exponential models. Unlike the Naive Bayes classifier, the Max Entropy does not assume that the features are conditionally independent of each other. The MaxEnt is based on the principle of maximum entropy and from all the models that fit our training data, selects the one which has the largest entropy.

The Maximum Entropy classifier is used when we can't assume the conditional independence of the features. This is particularly true in text classification problems where our features are usually words which obviously are not independent. The Max Entropy requires more time to train comparing to Naive Bayes, primarily due to the optimization problem that needs to be solved in order to estimate the parameters of the model. Nevertheless, after computing these parameters, the method provides robust results and it is competitive in terms of CPU and memory consumption.

Sentiment based classification of text documents is a more challenging tasks than topic based classification. Discrimination based on opinions, feelings, and attitudes is a more complex task than classification based on topics. The opinionated or subjective text on the Web is often non-structured or semi-structured from which feature selection is a crucial problem [25][26][27][28].

The target is to use the contextual information of the text (unigrams, bigrams, other characteristics within it) in order to categorize it to a given class (positive/neutral /negative). Following the standard bag-of-words framework that is commonly used in natural language processing and information retrieval, let $\{w_1, w_2, ..., w_n\}$ be the 'n' words that can appear in any given tweet. Then each tweet is represented by a sparse array with 1s and 0s that indicate whether a particular word w_i exists or not in the context of the tweet. This approach was proposed by [2].

The first step of constructing this model is to collect a large number of training data which consists of samples represented on the following format: (x_i, y_i) where x_i includes the contextual information of the document (the sparse array) and y_i its class. The second step is to summarize the training sample in terms of its empirical probability distribution:

$$\tilde{p}(x,y)=\frac{1}{N}\,\textit{number of times that}\,(x,y)\,\textit{occurs in the sample} \qquad [5.2a]$$

where N is the size of the training dataset.

We will use the above empirical probability distribution calculated from [5.2a] in order to construct the statistical model of the random process which assigns texts to a particular class by taking into account their contextual information. The building blocks of our model will be the set of statistics that come from our training dataset i.e. the empirical probability distribution.

We introduce the following indicator function:

$$f_i(x,y)=\begin{cases} 1 \textit{ if } y=c_i \textit{ and } x \textit{ contains } w_k \\ 0 \textit{ otherwise} \end{cases} \qquad [5.2b]$$

We call the above indicator function as "feature". This binary valued indicator function returns 1 only when the class of a particular document is c_i and the document contains the word w_k.

We express any statistic of the training dataset as the expected value of the appropriate binary-valued indicator function f_j. Thus the expected value of feature f_j with respect to the empirical distribution $\tilde{p}(x,y)$ is equal to:

$$\tilde{p}(f_j)=\sum_{x,y}\tilde{p}(x,y)f_j(x,y) \qquad [5.2c]$$

If each training sample (x,y) occurs once in training dataset then $\tilde{p}(x,y)$ is equal to ***1/N***.

When a particular statistic is useful to our classification, we require our model to accord with it. To do so, we constrain the expected value that the model assigns to the expected value of the feature function f_j. The expected value of feature f_j with respect to the model $p(y\,|\,x)$ is equal to:

$$p(f_j)=\sum_{x,y}\tilde{p}(x)p(y\,|\,x)f_j(x,y) \qquad [5.2d]$$

Equation [5.2d] is called constrain and we have as many constrains as the number of j feature functions.

The above constrains can be satisfied by an infinite number of models. So in order to build our model, we need to select the best candidate based on a specific criterion. According to the principle of Maximum Entropy, we should select the model that is as close as possible to uniform. In order words, we should select the model p^* with Maximum Entropy:

$$p^*=\arg\max_{p\subset C}\left(-\sum_{x,y}\tilde{p}(x)p(y\,|\,x)\log p(y\,|\,x)\right) \qquad [5.2e]$$

Given that:

1. $p(y \mid x) \geq 0 \text{ for all } x, y$ [5.2f]

2. $\sum_y p(y \mid x) = 1 \text{ for all } x$ [5.2g]

3. $\sum_{x,y} \tilde{p}(x) p(y \mid x) f_j(x,y) = \sum_{x,y} \tilde{p}(x,y) f_j(x,y) \text{ for } j \in \{1,2,\ldots n\}$ [5.2h]

To solve the above optimization problem we introduce the Lagrangian multipliers, we focus on the unconstrained dual problem and we estimate the lamda free variables $\{\lambda_1, \ldots, \lambda_n\}$ with the Maximum Likelihood Estimation method.

It can be proven that if we find the $\{\lambda_1, \ldots, \lambda_n\}$ parameters which maximize the dual problem, the probability given a document x to be classified as y is equal to:

$$p^*(y \mid x) = \frac{exp\left(\sum_t \lambda_t f_t(x,y)\right)}{\sum_y exp\left(\sum_t \lambda_t f_t(x,y)\right)} \qquad [5.2j]$$

Thus given that we have found the lamda parameters of our model, all we need to do in order to classify a new document is use the “maximum a posteriori” decision rule and select the category with the highest probability. Estimating the lamda parameters requires using an iterative scaling algorithm such as the GIS (Generalized Iterative Scaling) or the IIS (Improved Iterative Scaling).

The pseudo code for training one iteration of a GIS MaxEnt classifier as mentioned in [33] is given :

```
maxEntClassifyTrain():
$expected[0..F] = 0$
for each training instance $j$
    for each output $y$
        $s[j,y] = 0$
        for each $i$ such that $f_i(\bar{x}_j, y) \neq 0$
            $s[j,y] += \lambda_i \times f_i(\bar{x}_j, y)$
    $z = \sum_y e^{s[j,y]}$
    for each output $y$
        for each $i$ such that $f_i(\bar{x}_j, y) \neq 0$
            $expected[i] += f_i(\bar{x}_j, y) \times e^{s[j,y]} / z$
for each $i$
```

$$\delta_i = \frac{1}{f^{\#}} log\left(\frac{observed[i]}{exp\,ected[i]}\right)$$

$$\lambda_i + = \delta_i$$

5.3 Support Vector Machines :

In machine learning, support vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection [43].

Support vector machines (SVMs) are a range of classification and regression algorithms that have been based on the Structural Risk Minimization principle from statistical learning theory formulated by Vpnik [29][30][31].

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

In classification, items are represented by their features. In our case, tweets are represented by their words, so we will use words as features.

Scikit-learn [44] provides several vectorizers to translate the input documents into vectors of features (or feature weights). Typically we want to give appropriate weights to different words, and TF-IDF is one of the most common weighting schemes used in text analytics applications. In scikit-learn, we can use the TfidfVectorizer. The first call to fit_transform() will create the vocabulary (i.e. the list of words/features) and the feature weights from the training data. Secondly, we call simply transform() on the test data, which will create the feature weights for the test data, using the same vocabulary as the training data.

Given some training data $\boldsymbol{D}$, a set of $\boldsymbol{n}$ points of the form

$$\boldsymbol{D} = \{(x_i, y_i) \mid x_i \in \Re^p, y_i \in \{-1,1\}\}_{i=1}^{n} \qquad [5.3a]$$

where the y_i is either 1 or −1, indicating x_i the class to which the point belongs. Each x_i is a $\boldsymbol{p}$ -dimensional real vector. We want to find the maximum-margin hyperplane that divides the points having $y_i = 1$ from those having $y_i = -1$. Any hyperplane can be written as the set of points $\boldsymbol{x}$ satisfying

$$\boldsymbol{w \cdot x - b = 0} \qquad [5.3b]$$

where $\cdot$ denotes the dot product and $\boldsymbol{w}$ the (not necessarily normalized) normal vector to the hyperplane. The parameter $\frac{b}{\|w\|}$ determines the offset of the hyperplane from the origin along the normal vector $\boldsymbol{w}$.

If the training data are linearly separable, we can select two hyperplanes in a way that they separate the data and there are no points between them, and then try to maximize their distance. The region bounded by them is called "the margin". These hyperplanes can be described by the equations

$$\boldsymbol{w \cdot x - b = 1} \qquad [5.3c]$$

And

$$\boldsymbol{w \cdot x - b = -1} \qquad [5.3d]$$

Geometrically, the distance between these two hyperplanes is $\frac{2}{\|w\|}$, so to maximize the distance between the planes we want to minimize $\|w\|$. As we also have to prevent data points from falling into the margin, we add the following constraint: for each $\boldsymbol{i}$ either

$$w \cdot x_i - b \geq 1 \quad \textit{for } x_i \textit{ of the first class} \qquad [5.3e]$$

Or

$$w \cdot x_i - b \leq -1 \qquad \textit{for } x_i \textit{ of the second} \qquad [5.3f]$$

This can be rewritten as:

$$y_i(w \cdot x_i - b) \geq 1, \quad for\ all\ 1 \leq i \leq n. \qquad [5.3g]$$

We can put this together to get the optimization problem:

Minimize (in w, b)

$$\|w\|$$

subject to $(for\ any\ i = 1, \ldots, n)$

$$y_i(w \cdot x_i - b) \geq 1. \qquad [5.3h]$$

CHAPTER - 6

RESULTS AND CONCLUSION

6.1 Classifier Evaluation Metrics :

When we use a classifier model for a problem, we almost always want to look at the accuracy of that model as the number of correct predictions from all predictions made. This is the classification accuracy. When we need to decide whether it is a good enough model to solve the problem, accuracy is not the only metric for evaluating the effectiveness of a classifier. Two other useful metrics are precision and recall. These two metrics can provide much greater insight into the performance characteristics of a classifier.

A false positive error, or in short false positive, commonly called a 'false alarm', is a result that indicates a given condition has been fulfilled, when it actually has not been fulfilled i.e. erroneously a positive effect has been assumed.

A false negative error, or in short false negative, is where a test result indicates that a condition failed, while it actually was successful i.e. erroneously no effect has been assumed.

True positives are relevant items that we correctly identified as relevant.True negatives are irrelevant items that we correctly identified as irrelevant.

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

$$\boldsymbol{P = \frac{T_p}{T_p + F_p}} \qquad [6.1a]$$

where

$\boldsymbol{T_p}$ is the number of true positives

$\boldsymbol{F_p}$ is the number of false positives

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

$$\boldsymbol{R = \frac{T_p}{T_p + F_n}} \qquad [6.1b]$$

where

$\boldsymbol{T_p}$ is the number of true positives.

$\boldsymbol{F_n}$ is the number of false negatives.

Precision and recall can be combined to produce a single metric known as F-measure, which is the weighted harmonic mean of precision and recall.

$$\boldsymbol{F1 = 2\frac{P \times R}{P + R}} \qquad [6.1c]$$

where

$\boldsymbol{P}$ is the precision.

$\boldsymbol{R}$ is the recall.

We compare the performance of the three models that we use based on their precision, recall and F-measure.

6.2 Results :

The classifiers used take the training set of tweets as an input and thus learn from the them. The composition of each class of tweets of the training data is listed in Table 6.1.

Each of the classifiers used output the polarity of the sentiment of the tweets that they accept as an input. Thus, the sentiments of all the tweets corresponding to the given classifier are shown in Table 6.1 and Figure 6.1.

Polarity	Naive Bayes + Unigrams	Naive Bayes + Bigrams	MaxEnt + Unigrams	MaxEnt + Bigrams	Support Vector Machines
Positive	500	208	476	198	401
Negative	388	189	440	199	486
Neutral	180	671	152	671	181

Table 6.1 : Classification of Test Data

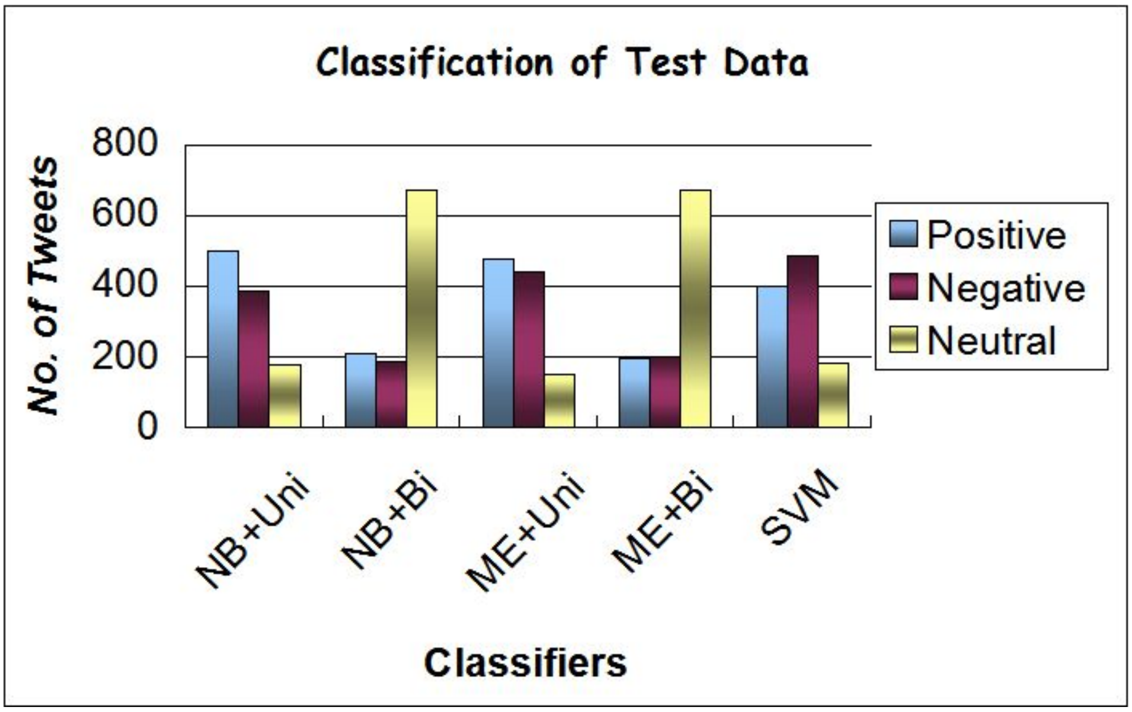


Figure 6.1 : Classification of Test Data

The performance of the classifiers is measured based on their respective precisions, recall value and F-measures.

The comparison of the classifiers for the Positive Class is shown in Table 6.2 and Figure 6.2.

Classifiers	POSITIVE CLASS		
	Precision	Recall	F-Measure
Naive Bayes+Unigrams	0.81	0.25	0.36
Naive Bayes+Bigrams	0.83	0.53	0.46
MaxEnt+Unigrams	0.85	0.50	0.76
MaxEnt+Bigrams	0.75	0.89	0.42
Support Vector Machines	0.81	0.69	0.76

Table 6.2 : Performance of Classifiers Used for Positive Class

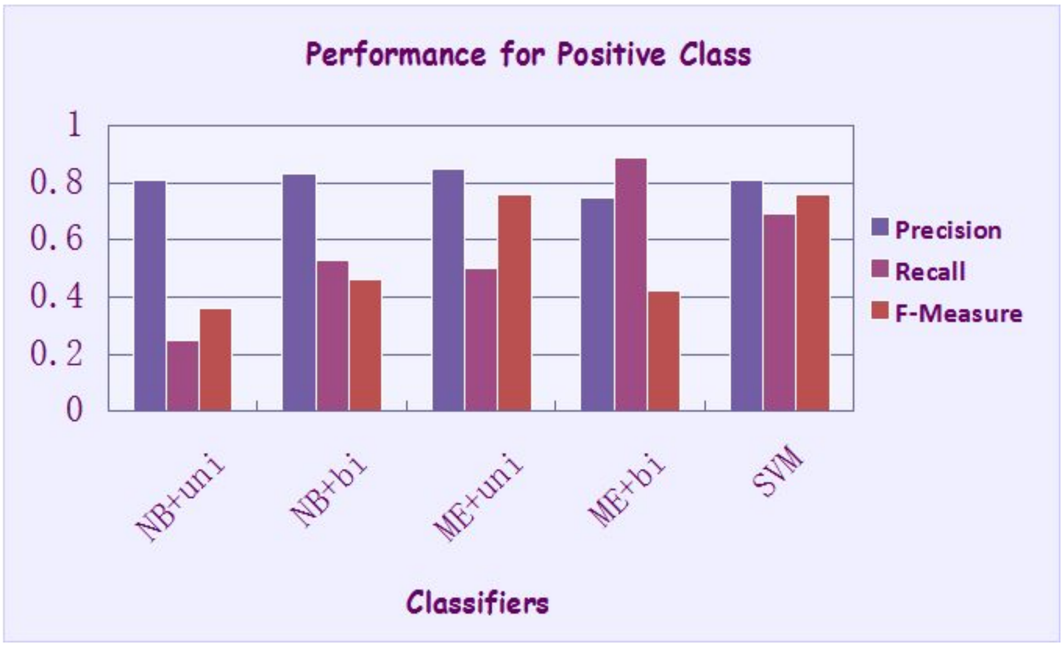


Figure 6.2 : Performance of Classifiers Used for Positive Class

The comparison of the classifiers for the Negative Class is shown in Table 6.3 and Figure 6.3.

Classifiers	NEGATIVE CLASS		
	Precision	Recall	F-Measure
Naive Bayes+Unigrams	0.79	0.65	0.37
Naive Bayes+Bigrams	0.76	0.63	0.88
MaxEnt+Unigrams	0.46	0.64	0.34
MaxEnt+Bigrams	0.74	0.54	0.88
Support Vector Machines	0.62	0.65	0.42

Table 6.3 : Performance of Classifiers Used for Negative Class

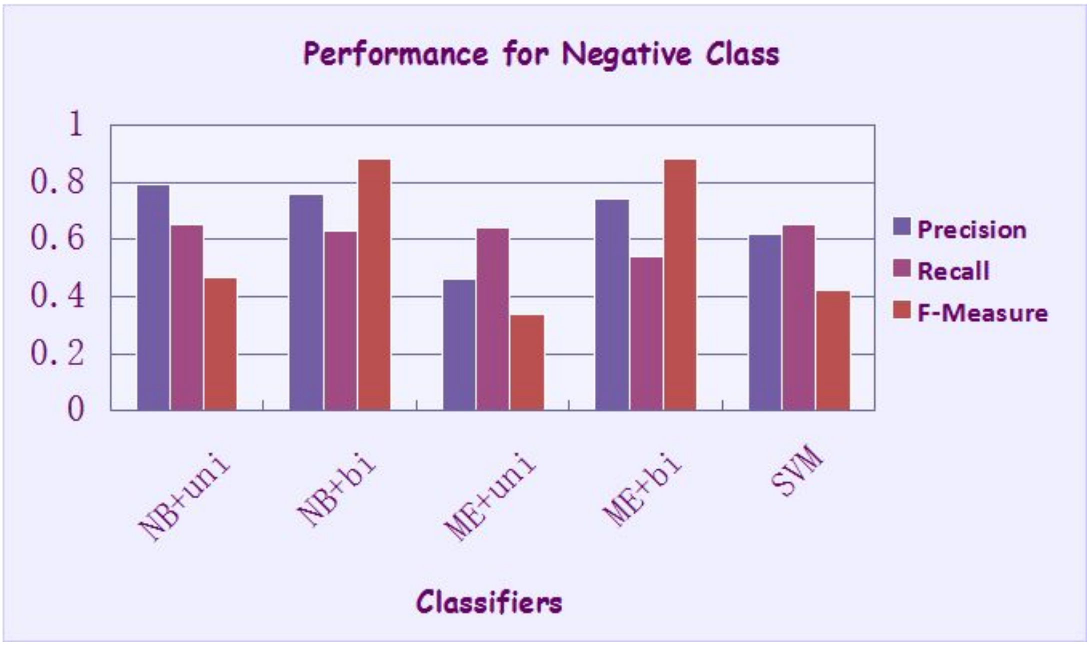


Figure 6.3 : Performance of Classifiers Used for Negative Class

The comparison of the classifiers for the Neutral Class is shown in Table 6.4 and Figure 6.4.

Classifiers	NEUTRAL CLASS		
	Precision	Recall	F-Measure
Naive Bayes+Unigrams	0.56	0.62	0.59
Naive Bayes+Bigrams	0.79	0.85	0.64
MaxEnt+Unigrams	0.66	0.62	0.64
MaxEnt+Bigrams	0.78	0.68	0.63
Support Vector Machines	0.33	0.28	0.25

Table 6.4 : Performance of Classifiers Used for Neutral Class

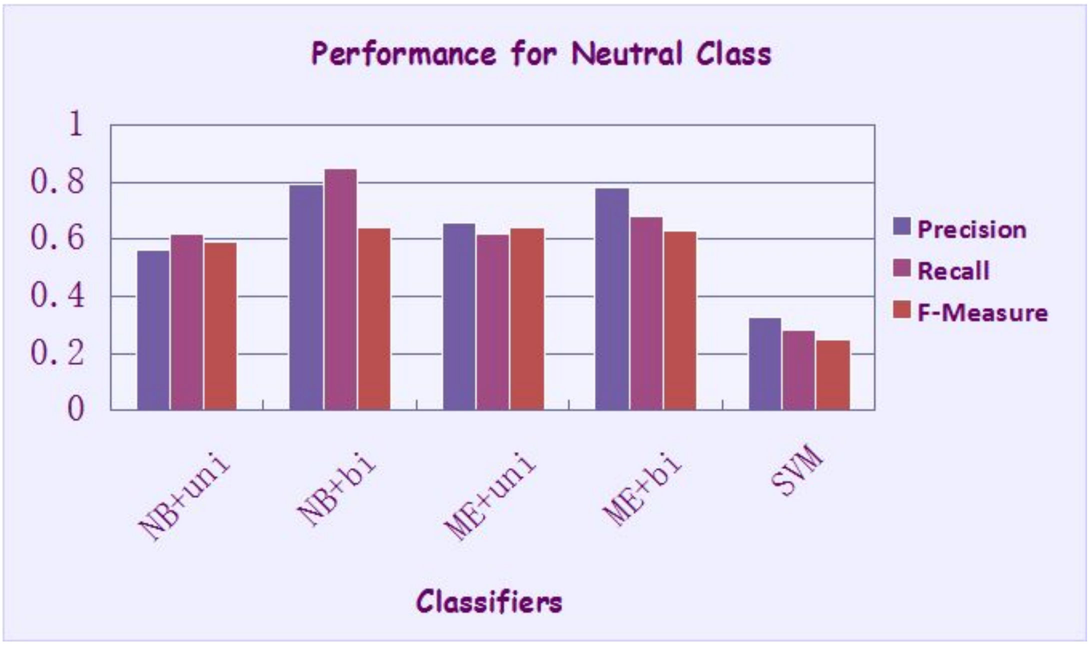


Figure 6.4 : Performance of Classifiers Used for Neutral Class

6.3 Conclusion:

Sentiment Classification has become a fascinating research area due to the availability of a huge volume of user-generated content in review sites, forums and blogs. The task of sentiment analysis, especially in the domain of micro-blogging, is still in the developing stage and far from complete. With the help of sentiment analysis, companies can estimate the extent of product acceptance and can devise strategies to improve their product. Individuals can also use opinion mining tools to make decisions on their buying by comparing competitive products not just based on specifications but also based on user experience and public opinions.

Right now we have worked with only the unigram and bigram based Naive Bayes and Maximum Entropy and the linear Support Vector Machine classifiers. As reported in the literature review section when bigrams are used along with unigrams this usually enhances performance. One more feature we that is worth exploring is whether the information about relative position of word in a tweet has any effect on the performance of the classifier. Many times we see customers posting the reviews in the blogs or forums with a lot of spelling mistakes which our dictionary cannot catch them and resulting in less accuracy of desired output. Most times of the times we see many promotions and spam tweets posted by the users. If we consider these tweets for performing sentiment analysis, we may get deviated from our desired results. So, lot of work has to be done in this field for identifying spam blogs, considering spelling mistakes and for other challenges.

REFERENCES:

[1] Bing, L. (2012). “Sentiment analysis: A fascinating problem.” *In Sentiment Analysis and Opinion Mining*, pages 7–143. Morgan and Claypool Publishers.

[2] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. “Thumbs up?: sentiment classification using machine learning techniques.” *Proceedings of the ACL-02 conference on Empirical methods in natural language processing* - Volume 10. Association for Computational Linguistics, 2002.

[3] Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. “Recognizing contextual polarity in phrase-level sentiment analysis.” *Proceedings of the conference on human language technology and empirical methods in natural language processing.* Association for Computational Linguistics, 2005.

[4] Turney, Peter D. “Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews.” *Proceedings of the 40th annual meeting on association for computational linguistics.* Association for Computational Linguistics, 2002.

[5] Brody, Samuel, and Nicholas Diakopoulos.“Cooooooooooooooollllllllllllll!!!!!!!!!!!!!!: using word lengthening to detect sentiment in microblogs.” *Proceedings of the conference on empirical methods in natural language processing.* Association for Computational Linguistics, 2011.

[6] Pak, Alexander, and Patrick Paroubek. “Twitter as a Corpus for Sentiment Analysis and Opinion Mining.” *LREC.* Vol. 10. 2010.

[7] Barbosa, Luciano, and Junlan Feng. “Robust sentiment detection on twitter from biased and noisy data.” *Proceedings of the 23rd International Conference on Computational Linguistics: Posters.* Association for Computational Linguistics, 2010.

[8] Kouloumpis, Efthymios, Theresa Wilson, and Johanna Moore. “Twitter sentiment analysis: The good the bad and the omg!.” *Icwsm* 11 (2011): 538-541.

[9] Go, Alec, Richa Bhayani, and Lei Huang. “Twitter sentiment classification using

distant supervision.” *CS224N Project Report, Stanford* 1 (2009): 12.

[10] Baccianella, Stefano, Andrea Esuli, and Fabrizio Sebastiani. “SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining.” *LREC*. Vol. 10. 2010.

[11] Prabowo, Rudy, and Mike Thelwall. “Sentiment analysis: A combined approach.” *Journal of Informetrics 3.2* (2009): 143-157.

[12] Bollen, Johan, Alberto Pepe, and Huina Mao. “Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena.” *arXiv preprint arXiv:0911.1583* (2009).

[13] Tumasjan, Andranik, et al. “Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment.” *ICWSM 10* (2010): 178-185.

[14] Dwi Prasetyo, Nugroho, and Claudia Hauff. “Twitter-based Election Prediction in the Developing World.” *Proceedings of the 26th ACM Conference on Hypertext & Social Media*. ACM, 2015.

[15] Almatrafi, Omaima, Suhem Parack, and Bravim Chavan. “Application of location-based sentiment analysis using Twitter for identifying trends towards Indian general elections 2014.” *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication*. ACM, 2015.

[16] Salton, Gerard, and Michael J. McGill. “Introduction to modern information retrieval.” (1986).

[17] Taboada, Maite, et al. “Lexicon-based methods for sentiment analysis .” *Computational linguistics* 37.2 (2011): 267-307.

[18] Rish, Irina. “An empirical study of the naive Bayes classifier.” *IJCAI 2001 workshop on empirical methods in artificial intelligence*. Vol. 3. No. 22. IBM New York, 2001.

[19] Langley, Pat, Wayne Iba, and Kevin Thompson. “An analysis of Bayesian classifiers.” *AAAI*. Vol. 90. 1992.

[20] Friedman, Nir, Dan Geiger, and Moises Goldszmidt. “Bayesian network classifiers.” *Machine learning* 29.2-3 (1997): 131-163.

[21] Domingos, Pedro, and Michael Pazzani. "On the optimality of the simple Bayesian classifier under zero-one loss." *Machine learning* 29.2-3 (1997): 103-130.

[22] Lewis, David D. "Naive (Bayes) at forty: The independence assumption in information retrieval." *Machine learning: ECML-98*. Springer Berlin Heidelberg, 1998. 4-15.

[23] Cohen, William W., and Yoram Singer. "Context-sensitive learning methods for text categorization." *ACM Transactions on Information Systems (TOIS)* 17.2 (1999): 141-173.

[24] Wang, Ye-Yi, and Alex Acero. "Maximum entropy model parameterization with TF∗ IDF weighted vector space model." *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*. IEEE, 2007.

[25] Sharma, Anuj, and Shubhamoy Dey. "A comparative study of feature selection and machine learning techniques for sentiment analysis." *Proceedings of the 2012 ACM Research in Applied Computation Symposium*. ACM, 2012.

[26] Eirinaki, Magdalini, Shamita Pisal, and Japinder Singh. "Feature-based opinion mining and ranking." *Journal of Computer and System Sciences* 78.4 (2012): 1175-1184.

[27] Wang, Suge, et al. "A feature selection method based on improved fisher's discriminant ratio for text sentiment classification." *Expert Systems with Applications* 38.7 (2011): 8696-8702.

[28] Zhai, Zhongwu, et al. "Exploiting effective features for chinese sentiment classification." *Expert Systems with Applications* 38.8 (2011): 9139-9146.

[29] Vapnik, Vladimir. "The nature of statistical learning theory." Springer Science & Business Media, 2013.

[30] Cristianini, Nello, and John Shawe-Taylor. "An introduction to support vector machines and other kernel-based learning methods." Cambridge university press, 2000.

[31] Li, Kun-Lun, et al. "Active learning with simplified SVMs for spam categorization." *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*. Vol. 3. IEEE, 2002.

[32] Yang, Ying, and Geoffrey I. Webb. "A comparative study of discretization methods for naive-bayes classifiers." *Proceedings of PKAW*. Vol. 2002. 2002.

[33] Goodman, Joshua. "Sequential conditional generalized iterative scaling." *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2002.

[34] Twitter *https://twitter.com/*

[35] Flipkart *http://www.flipkart.com/*

[36] Multi Perspective Question Answering (MPQA) Online Lexicon *http://www.cs.pitt.edu/mpqa /subj_lexicon.html*

[37] Twitter Sentiment, an online application performing sentiment classification of Twitter. *http://twittersentiment.appspot.com/*

[38] Tweepy *http://www.tweepy.org/*

[39] GitHub *https://github.com/*

[40] Python *https://www.python.org/*

[41] Twitter API *https://dev.twitter.com/overview/documentation*

[42] Natural Language Toolkit *http://www.nltk.org/*

[43] Support Vector Machines using Scikit-learn *http://scikit-learn.org/stable/modules /svm.html#svm*

[44] Scikit-learn *http://scikit-learn.org/*