Chapter 1

Introduction

1.1 Text Mining and corpus

Text mining or text analytics is a method for drawing out content based on meaning and context from a large body (or bodies) of text. It is a method for gathering structured information from unstructured text. So, Text mining, basically, refers to the analysis of large quantities of natural language text to identify lexical or linguistic usage patterns of interest, and thus extracting potentially useful information. But the challenge is that unlike data stored in databases, text that is to be mined for insights is unstructured and amorphous, making it difficult for us to process algorithmically.

Within the World Wide Web, micro blogging has become a very popular communication tool for internet users where they share opinions on different aspects of their day-to-day life. People share opinions on variety of topics and discusses current issues making micro blogging web-sites valuable sources of people's opinions and sentiments as more and more users post about products and services they use, movies they have watched, or express their political and religious views.

For our work, we have used Twitter and NLTK corpuses as our datasets for sentiment analysis. We have chosen Twitter because Twitter's audience varies from regular users to celebrities, company representatives, politicians, and even country presidents. Therefore, it is possible to collect text posts of users from different social and interests groups.

We have chosen NLTK as NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. Moreover, NLTK is available for Windows, Mac OS X, and Linux and best of all, NLTK is a free, open source, community-driven project.

1.2 Sentiment Analysis

Sentiment Analysis (SA) is the process of determining whether a piece of writing is positive or negative. A sentiment analysis system for text analysis combines natural language processing (NPL) and machine learning techniques to assign weighted sentiment scores to the entities, topics, themes and categories within a sentence or phrase.

Basic sentiment analysis of the text documents follows a straightforward process:

- Break each text document into its component parts (Sentences ,phrases ,tokens and parts of speech).
- Identify each sentiment-bearing phrase and component.

- Assign a sentiment score to each phrase and component (-1 or +1).

The first step in the sentiments classification problem is to extract and select text features. Some of the current features are Terms presence and frequency. These features are individual words or word n-grams and their frequency counts. It either gives the words binary weighting (zero if the word appears or one if otherwise) or uses term frequency weights to indicate the relative importance of features. Parts of speech (POS) means finding adjectives, as they are important indicators of opinions. Opinion words and phrases, these words are commonly used to express opinions including good or bad, like or hate. On the other hand, some phrases express opinions without using opinion words. For example: cost me an arm and a leg. Negations are the appearance of negative words that may change the opinion orientation like not well is equivalent to bad.

Now, Sentiment Classification techniques can be roughly divided into machine learning approach, lexicon-based approach and hybrid approach. Sentiment Classification techniques can be roughly divided into machine learning approach, lexicon-based approach and hybrid approach. The Machine Learning Approach (ML) applies the famous ML algorithms and uses linguistic features. The Lexicon-based Approach relies on a sentiment lexicon, a collection of known and precompiled sentiment terms. It is divided into dictionary-based approach and corpus-based approach which use statistical or semantic methods to find sentiment polarity. The various approaches and the most popular algorithms of SC are illustrated in Fig1.1. The text classification methods using ML approach can be roughly divided into supervised and unsupervised learning methods. The supervised methods make use of a large number of labeled training documents. The unsupervised methods are used when it is difficult to find these labeled training documents. The lexicon-based approach depends on finding the opinion lexicon which is used to analyze the text. There are two methods in this approach. The dictionary-based approach which depends on finding opinion seed words, and then searches the dictionary of their synonyms and antonyms. The corpus-based approach begins with a seed list of opinion words, and then finds other opinion words in a large corpus to help in finding opinion words with context specific orientations. This could be done by using statistical or semantic methods.

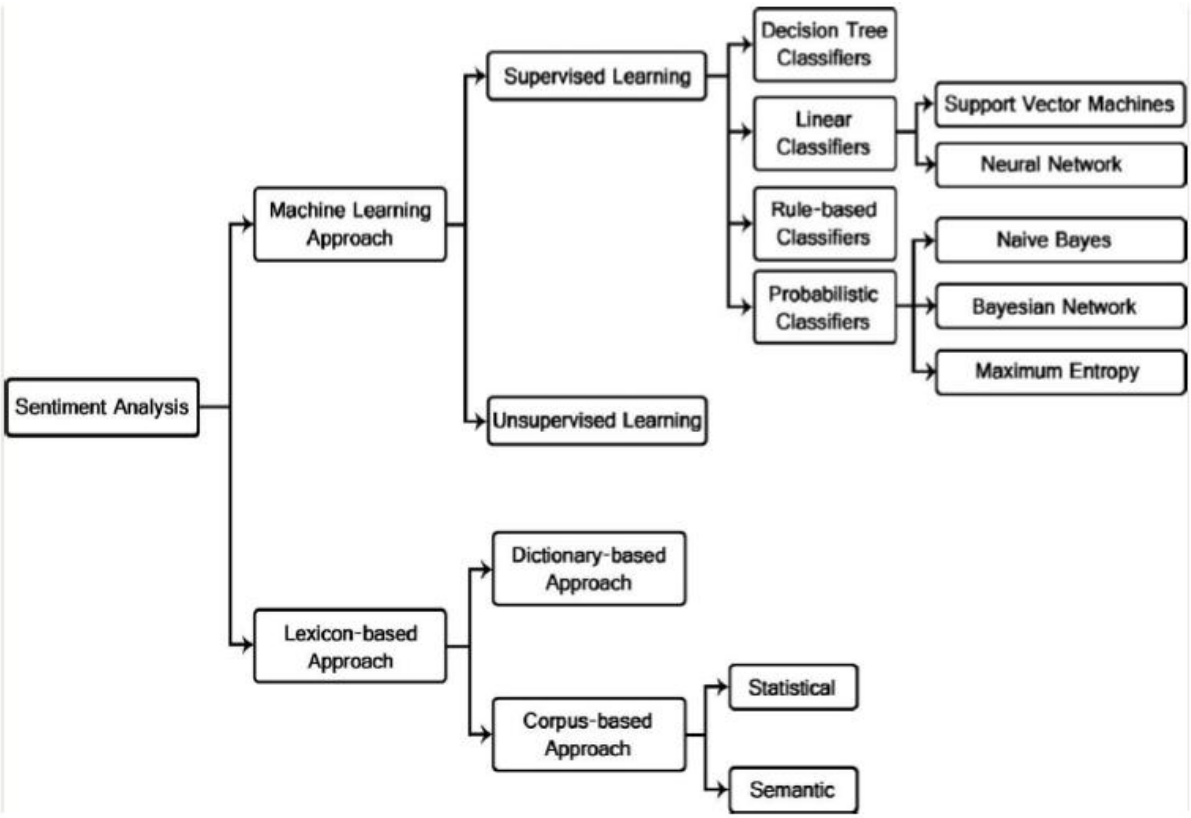


Fig 1.1: Sentiment classification techniques.

1.3 Classifiers

1.3.1 Naïve Bayes Classifiers

Bayes' theorem is the basis for Naive Bayes classifier. To be more specific Naive Bayes is an extended version of Bayes' Theorem, which is capable of handling classification in case of multiple data points. The power of Naive Bayes comes from its assumption that each data points or feature is independent of the other. Bayes' theorem is actually used to find the probability of an event, based upon prior knowledge. Say, H is a hypothesis and E be evidence, then according to Bayes' theorem

$$P(\frac{H}{E}) = \frac{P(\frac{E}{\mathrm{H}}) \times P(H)}{P(E)}$$

Where, P(H) represents probability of hypothesis before getting the evidence i.e., prior probability, P(H/E) represents probability of hypothesis after getting the evidence i.e., posterior probability.

Now the problem with Bayes' theorem is its limited capability of classification, which bounds it to only one feature. But Naive Bayes' is capable to classify even when we have multiple features, as it assumes that each data point is independent.

$$P(\frac{H}{E1, E2, ..., En}) = \frac{P(H) \times P(\frac{E1}{H}) \times ... \times P(\frac{E1}{H})}{P(E1, E2, ..., En)}$$

1.3.2 Multinomial Naïve Bayes

Multinomial Naive Bayes classifier is a specific instance of a Naive Bayes classifier which uses a multinomial distribution for each of the features. Feature vectors represent the frequencies with which certain events have been generated by a multinomial distribution. This is the event model typically used for document classification.

1.3.3 Bernoulli Naïve Bayes

In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence(i.e. a word occurs in a document or not) features are used rather than term frequencies(i.e. frequency of a word in the document).

1.3.4 Support Vector Machine

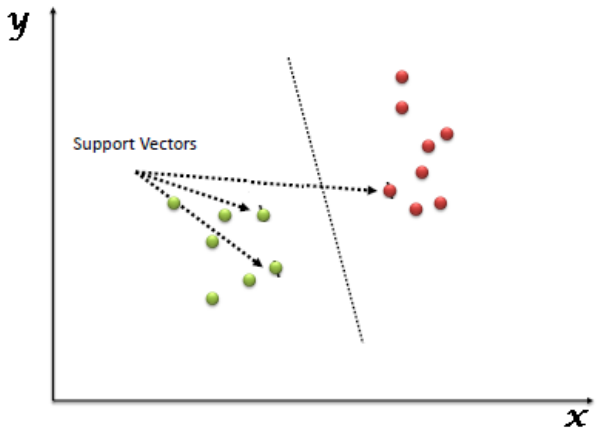


Fig 1: SVM

In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features we have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well. Support Vectors are simply the co-ordinates of individual observation. Functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels.

1.3.5 Logistic Regression

Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. That is instead of predicting exactly 0 or 1, logistic regression generates a probability, a value between 0 and 1, exclusive.

1.4 Tweepy

Twitter scraping is the act of copying a twitter post. Tweepy is a set of tools (code) that let us access Twitter through the API (Application Program Interface) directly. Tweepy has more capability for programmatically accessing and parsing Twitter feeds/tweets than simple scraping. It is very useful for large volumes of tweets.

To access the Twitter API, we need 4 things from the our Twitter App page (which we need to build). These keys are located in our Twitter app settings in the Keys and Access Tokens tab.

- consumer key
- consumer seceret key
- access token key
- access token secret key

We should not share these with anyone else because these values are specific to our app and we should take of this fact seriously. Once we have these keys we need to perform some authentication after which we will be able to search for tweets based upon queries. The below query is used to search and fetch tweets.

```
API.search(q[, lang][, locale][, rpp][, page][, since_id][, geocode][, show_user])
```

Parameters:

q – the search query string
lang – Restricts tweets to the given language, given by an ISO 639-1 code.
locale – Specify the language of the query we are sending.
rpp – The number of tweets to return per page, up to a max of 100.
page – The page number (starting at 1) to return, up to a max of roughly 1500 results
since_id – Returns only statuses with an ID greater than (that is, more recent than) the specified ID.
geocode – Returns tweets by users located within a given radius of the given latitude/longitude.
show_user –This is useful for readers that do not display Atom's author field. The default is false.

1.5 Aylienapi Client

Once we collected the text of the Tweets that we want to analyse, we can use more advanced NPL tools to start extracting information from. For sentiment analysis, we can use AYLIEN Text API. Just like with the Twitter Search API, here also we need to sign up for the free plan to grab this API. This plan gives us 1,000 calls to the API per month free of charge. They provides services up to some certain time, but after some duration they generally ask for payment .

1.6 Matplotlib

In our project we actually required a lot of graphs to express our results. So for the purpose of plotting we have used matplotlib. Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.

Chapter 2

Literature Survey

We started the survey by reading various research papers. To determine the sentiment polarity, some researchers proposed a novel machine-learning method that applies text-categorization techniques to just the subjective portions of the document and removing the objective portion of the document. The removal has been achieved using efficient techniques for finding 'minimum cuts in graph[1].

As mentioned that twitter has become a very popular communication tool for Internet users to share opinions on various aspects of everyday life. There were some researchers who started by collecting corpus on the basis of happy emoticons: ":-)", ":)", "=)", ":D" etc. and sad emoticons: ":-(", ":(", "=(", ";(" etc. To analyze the subjectivity and objectivity they have used POS tags like, utterances (UH), simple past tense (VBD), etc. are the indicator of a subjective text whereas presence of common and proper nouns (NPS, NP, NNS), mostly indicates objective sentences[2].

Mohamed M. Mostafe has used an expert-predefined lexicon including around 6800 seed adjectives with known orientation to conduct the analysis. The sentiment has been determined by comparing tweets against the expert-defined entry in the dictionary, which makes it easy to determine the polarity of a specific sentence [3].

Ronen Feldman has discussed various sentiment analysis techniques. As per him, document-level sentiment analysis (where we assign one sentiment to the whole document), sentence-level sentiment analysis (here each document is having multiple sentences having different opinions) and the last aspect-based sentiment analysis are the three kind of sentiment analysis that we do with documents. In many cases people talk about entities that have many aspects (attributes) and they have a different opinion about each of the aspects. For e.g., "As a long-time Kindle fan I was eager to get my hands on a Fire. There are some great aspects; the device is quick and for the most part dead-simple to use. The screen is fantastic with good brightness and excellent color, and a very wide viewing angle. But there are some downsides too; the small bezel size makes holding it without inadvertent page-turns difficult, the lack of buttons makes controls harder, the accessible storage memory is limited to just 5GB." So, basically we will be working with sentence level document analysis[5].

There were some works which was based on lexicon based approach. Like the one presented by Stefano Baccianella et. al. , the SENTIWORDNET 3.0, a lexical resource explicitly devised for supporting sentiment classification and opinion mining applications which is the result of automatically annotating all WORDNET synsets according to their degrees of positivity, negativity, and neutrality[6].

There were some lexicon based approach for sentiment classification of Twitter posts, based on the exploitation of widespread lexical resources such as SentiWordNet, WordNet-Affect, MPQA and SenticNet[8].

Lexicon-based approaches for sentiment classification are based on the insight that the polarity of a piece of text can be obtained on the ground of the polarity of the words which compose it. But so simple approach is likely to fail, so T.Meyyappan et. al. modified the approach as per which, given a Tweet T, they split it in several micro-phrases m1 : : :mn according to the splitting cues occurring in the content [10]. As splitting cues they have used punctuations, adverbs and conjunctions. Whenever a splitting cue is found in the text, a new micro-phrase is built. Hence the sentiment S conveyed by a Tweet T is defined as the sum of the polarity conveyed by each of the micro-phrases mi which compose it. When a negation has been found in the text, the polarity of the whole micro-phrase is inverted.

When it comes to machine learning approach we need to define features based upon which we need to do classification. There are different types of features like, Hyperbole features are defined as those where there is a presence of the intensified positive words(adjectives), interjections, quotes, punctuation marks. Pragmatic features are those where there is the presence of emoticons like frowning smiley, smiling faces etc and the mentions in the comments or the replies in case of twitter re-tweets.

There were papers that talked about different kinds of features that we can select for the task of classification. The features are like terms presence and frequency (BOW), parts of speech (POS), opinion words and phrases and negation. The feature selection treat the document either as a group of words (Bag of Words (BOWs)) or as a string which retains the sequence of words in the document. They have used various statistical methods like Point-wise Mutual Information (PMI), Chi- square and latent Semantic Indexing (LSI) [9, 12].

But we emphasize on the importance of handling negation. Negation is a very common linguistic construction that affects polarity and therefore, needs to be taken into consideration in sentiment analysis. The challenge is that negation is not only conveyed by common negation words (not, neither, nor) but also by other lexical units. Research in the field has shown that there are many other words that invert the polarity of an opinion expressed, such as valence shifters, connectives or modals. “I find the functionality of the new mobile less practical”, is an example for valence shifter, “Perhaps it is a great phone, but I fail to see why”, shows the effect of connectives. An example sentence using modal is, “In theory, the phone should have worked even under water”. As can be seen from these examples, negation is a difficult yet important aspect of sentiment analysis [13, 14].

Now we should always remember the fact that in order to apply classifier, it is essential to pre-process or clean the raw data. The preprocessing task that are being performed involves removal of stop-words from raw data, removal of affixes, removal of other notation like hash tag, @, RT, emoticons, URLs, convert acronyms[14]. Other than the above mentioned methods for data cleaning they have also suggested additionally the “Removal of Extra spaces”, “Emoticons used replaced with their actual meaning like Happy, Sad by using Emoticon data set available on Internet”, “Abbreviations like OMG, are replaced by their actual meanings”, “Pragmatics handling like hapyyyyyyy as happy, gudddddd as good”, Punctuations, Numbers etc[17].

The tools that we can use for sentiment classification have also been discussed like python NLTK, Opinion Finder, Apache OpenNLP, Web Fountain, WEKA, Ling Pipe, Opinion Observer, Review Seer Tool, Red Opals and Stanford Parser [16].
NLTK is a python library mainly used in Sentiment analysis for Tokenization, Stop Word removal, Stemming and tagging. General Architecture for Text Engineering (GATE) is

information Extraction System consisting of modules that Tokenizes, Stemming and Part of speech tagger. This tool is written in Java language. Red Opal is widely used for users who want to buy any products based on different features. Users can search for any product depending upon the feature selected and can get reviews related to their search. Opinion Finder is used for analysis of different Subjective sentences related to any topic & classification of sentences is done based on their polarity. It's written in Java and is platform Independent tool. Apache OpenNLP is a toolkit which is based on machine learning techniques. The most common NLP task included are tokenizer, part of speech tagger, named entity extractor, chunker, parser, and conference resolution. Web Fountain is a sentiment analysis tool that completes the requirements of analysis agents such as text gathering, storing indexing and querying. WEKA is based on machine learning techniques. JAVA programming language is used to implement this tool, and it has its GUI to show the data. Ling pipe is used for linguistic processing for text, including clustering, cataloging, and extraction. Stanford Parser is used as a POS tagger and sentence parsing from the NLP group.

Victoria Ikoro1 et. al. have successfully highlighted one of the challenges in sentiment analysis, that is the lack of annotated data sets that can be used to train a model capable of adjusting to differences in multiple domains [18]. But, creating a domain-specific dictionary can be very expensive and time-consuming. So they optimized the accuracy of sentiment analysis by combining functions from two sentiment lexica namely Sentimentr and the Hu & Liu opinion lexicon. They then saw Sentimentr performed well at detecting negative sentiments but struggle to discriminate between positive and negative tweets. On the other hand Hu and Liu opinion lexicon was better at detecting the positive and neutral tweets. Hence after passing the tweets through sentiment, they masked its negative and neutral classification and passed it through Hu and Lie opinion lexicon. Finally they combined the results from both lexica.

So, this was a brief literature survey that highlighted problems and some solutions to various sentiment analysis issues. Although using some complex methods researchers obtained very good accuracy but they requires either very complex calculation or high amount to training time like deep learning approaches. Different researchers have used different data cleaning processes and feature selection methods. So based upon all these observation we come up to the objectives of our project discussed in next chapter.

Chapter 3

Present Work

In our work we have taken Machine Learning Approach for sentiment analysis (the other being Lexicon based approach), and used different classifiers to predict the sentiment.

3.1 Objectives

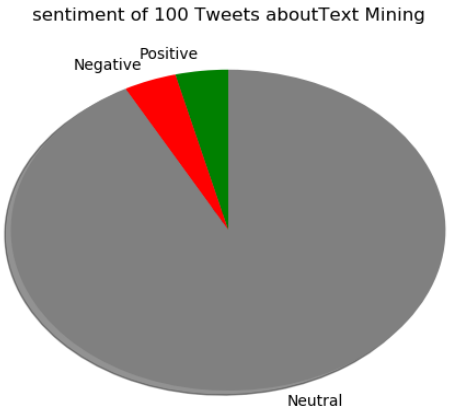
The objective of our project is to enhance the accuracy of sentiment classification of tweets. For this task we need to find:

- the best classifiers for this task
- the necessary data cleaning methods and
- proper feature extraction and feature selection

3.2 A work on Sentiment Analysis using aylienapiclient.

This task has been achieved using 3 steps.

- Using Tweepy building the corpus by gathering sample text data from Twitter's API.
- Then analysing the sentiment of a piece of text with aylienapiclient.
- Then use Pandas and matplotlib to visualize the results of our work.



Graph 1: output for Aylien api client.

So basically we have used Tweepy to ask the Twitter API for 500 of the most recent Tweets that contains our search term, and then we'll write the Tweets to a text file, with each Tweet on its own line. This makes it easy for us to analyze each Tweet separately in the next step.

Then we analyzed corpus collected using Tweepy, each tweets separately. For each Tweet in the .txt file we sent the text to the AYLIEN API, then extracted the sentiment prediction from the JSON that the AYLIEN API returns, and writes this to the .csv file along with the Tweet itself. This gave us a .csv file with two columns — the text of a Tweet and the sentiment of the Tweet, as predicted by the AYLIEN API. Then we used the data to build informative visualizations using matplotlib, as shown in Graph 1. But using although we can classify, but we can never built our own module for sentiment classification. So, we took a new road, and started our journey with NLTK to achieve our objective of developing a python module for sentiment analysis.

3.3 Use of NLTK to perform sentiment analysis.

As already been discussed in the introduction what NLTK is and what are some of its implementations, here we will see the implementation part. NLTK helped us with everything from splitting sentences from paragraphs, splitting up words, recognizing the part of speech of those words, highlighting the main subjects, etc.

We have also performed a comparison among all of them to find out which one works best for sentiment classification. So, let's start with NLTK and its implementations:

3.2.1 Tokenizing words and sentences with NLTK

Let's see an example of how one might actually tokenize something into tokens with the NLTK module.

```
from nltk.tokenize import sent_tokenize, word_tokenize

EXAMPLE_TEXT = "Hello Mr. Smith, how are you doing today? The weather is great, and Python is awesome. The sky is pinkish-blue. You shouldn't eat cardboard."

print(sent_tokenize(EXAMPLE_TEXT))
```

'sent_tokenize' helps in tokenizing paragraph into sentences. So, output will be

['Hello Mr. Smith, how are you doing today?', 'The weather is great, and Python is awesome.', 'The sky is pinkish-blue.', "You shouldn't eat cardboard."]

```
print(word_tokenize(EXAMPLE_TEXT))
```

'word_tokenize' helps in tokenizing sentence into words. So, output will be

Output is: ['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The', 'weather', 'is', 'great', ',', 'and', 'Python', 'is', 'awesome', '.', 'The', 'sky', 'is', 'pinkish-blue', '.', 'You', 'should', "n't", 'eat', 'cardboard', '.']

3.2.2 Stopwords

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
for w in word_tokens:
   if w not in stop_words:
      filtered_sentence.append(w)
```

With the help of stopwords module and a loop we can filter out all the stopwords from a document.

3.2.3 Stemming words with NLTK

Stemming is the process of producing morphological variants of a root/base word. A stemming algorithm reduces the words “chocolates”, “chocolatey”, “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”.

```
from nltk.stem import PorterStemmer
ps = PorterStemmer()
```

So, for the purpose of stemming we have used one of the most popular stemming algorithms, Porter Stemmer.

3.2.4 POS tagging with NLTK

One of the most powerful aspects of the NLTK module is the Part of Speech tagging. This means labeling words in a sentence as nouns, adjectives, verbs...etc and sometimes also labeling by tense, and more. Here's a list of the tags,

```
POS tag list:

CC      coordinating conjunction
CD      cardinal digit
DT      determiner
EX      existential there (like: "there is" ... think of it like "there exists")
FW      foreign word
IN      preposition/subordinating conjunction
JJ      adjective 'big'
```

JJR	adjective, comparative 'bigger'
JJS	adjective, superlative 'biggest'
LS	list marker
MD	modal could, will
NN	noun, singular 'desk'
NNS	noun plural 'desks'
NNP	proper noun, singular 'Harrison'
NNPS	proper noun, plural 'Americans'
PDT	pre determiner 'all the kids'
POS	possessive ending parent\'s
PRP	personal pronoun I, he, she
PRP$	possessive pronoun my, his, hers
RB	adverb very, silently,
RBR	adverb, comparative better
RBS	adverb, superlativebest
RP	particle give up
TO	to go 'to' the store.
UH	interjection
VB	verb, base form take
VBD	verb, past tense took
VBG	verb, gerund/present participle taking
VBN	verb, past participle taken
VBP	verb, sing. Present, non-3d take
VBZ	verb, 3rd person sing. Present takes
WDT	wh-determiner which
WP	wh-pronoun who, what
WP$	possessive wh-pronoun whose
WRB	wh-abverb where, when

The below piece of code will add a tag with each word.

```
tagged = nltk.pos_tag(words)
```

3.2.5 The Corpora with NLTK

The NLTK corpus is a massive dump of all kinds of natural language data sets that are definitely worth taking a look at. These files are plain text files for the most part, some are XML and some are other formats, but they can all be accessed manually, or via the module and Python.

WordNet is a lexical database for the English language, which was created by Princeton, and is part of the NLTK corpus. We can also use WordNet alongside the NLTK module to find the meanings of words, synonyms, antonyms, and more.

So our first dataset was movie reviews database that is part of the NLTK corpus. The NLTK corpus movie_reviews data set has the reviews, and they are labeled already as positive or negative. Hence we have used it to train and test with the classifiers.

```
import nltk
import random
from nltk.corpus import movie_reviews

documents = [(list(movie_reviews.words(fileid)), category)
        for category in movie_reviews.categories()
        for fileid in movie_reviews.fileids(category)]

random.shuffle(documents)

all_words = []
for w in movie_reviews.words():
   all_words.append(w.lower())

all_words = nltk.FreqDist(all_words)

word_features = list(all_words.keys())[:5000]
```

word_features contains the top 5000 most common words. Then we have build a quick function that will find these top 5000 words in our positive and negative documents, marking their presence as either positive or negative:

```
def find_features(document):
   words = set(document)
   features = {}
   for w in word_features:
      features[w] = (w in words)

   return features
```

Finally to get the featureset, we have used the below list comprehension.

```
featuresets = [(find_features(rev), category) for (rev, category) in documents]
```

3.2.6 Naïve Bayes Classifier with NLTK

Before we can train and test the algorithm, we first need to split up the data into a training set and a testing set.

```
# set that we'll train our classifier with
training_set = featuresets[:1900]

# set that we'll test against.
testing_set = featuresets[1900:]
```

Then we just called the Naïve Bayes classifier and used .train() to train it using the training data as follows:

```
classifier = nltk.NaiveBayesClassifier.train(training_set)

# testing the test data
print("Classifier accuracy percent:",(nltk.classify.accuracy(classifier, testing_set))*100)
```

3.2.7 Sklearn with NLTK

The best module for Python to try new classifiers is the Scikit-learn (sklearn) module.

```
from nltk.classify.scikitlearn import SklearnClassifier

#couple more variations of the Naive Bayes algorithm
from sklearn.naive_bayes import MultinomialNB,BernoulliNB

MNB_classifier = SklearnClassifier(MultinomialNB())
MNB_classifier.train(training_set)
print("MultinomialNB accuracy percent:",nltk.classify.accuracy(MNB_classifier,
testing_set))

BNB_classifier = SklearnClassifier(BernoulliNB())
BNB_classifier.train(training_set)
print("BernoulliNB accuracy percent:",nltk.classify.accuracy(BNB_classifier, testing_set))
```

Then we incorporated some more classifiers, as follows:

```
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC

LogisticRegression_classifier = SklearnClassifier(LogisticRegression())
LogisticRegression_classifier.train(training_set)
print("LogisticRegression_classifier accuracy percent:",
(nltk.classify.accuracy(LogisticRegression_classifier, testing_set))*100)

SVC_classifier = SklearnClassifier(SVC())
SVC_classifier.train(training_set)
print("SVC_classifier accuracy percent:", (nltk.classify.accuracy(SVC_classifier,
testing_set))*100)

LinearSVC_classifier = SklearnClassifier(LinearSVC())
LinearSVC_classifier.train(training_set)
print("LinearSVC_classifier accuracy percent:",
(nltk.classify.accuracy(LinearSVC_classifier, testing_set))*100)

NuSVC_classifier = SklearnClassifier(NuSVC())
NuSVC_classifier.train(training_set)
```

```
print("NuSVC_classifier accuracy percent:", (nltk.classify.accuracy(NuSVC_classifier, testing_set))*100)
```

This way we have incorporated all the classifiers in our python module, which we are developing for classification of sentiments.

3.2.8 Importing new dataset (of tweets)

As our goal is to do Twitter sentiment analysis, so we're hoping for a data set that is a bit shorter per positive and negative statement. So we downloaded the data from internet as text files, consisting of positive and negative tweets, a total of 10,662 tweets equally divided into positive and negative tweets.

```
short_pos = open("short_reviews/positive.txt","r").read()
short_neg = open("short_reviews/negative.txt","r").read()

documents = []

for r in short_pos.split('\n'):
    documents.append( (r, "pos") )

for r in short_neg.split('\n'):
    documents.append( (r, "neg") )


all_words = []

short_pos_words = word_tokenize(short_pos)
short_neg_words = word_tokenize(short_neg)

for w in short_pos_words:
    all_words.append(w.lower())

for w in short_neg_words:
    all_words.append(w.lower())

all_words = nltk.FreqDist(all_words)

word_features = list(all_words.keys())[:5000]

#for this part we have applied k-folds to test and train using different sets at different times.
training_set = featuresets[:10000]
testing_set =  featuresets[10000:]
```

So, with the help of all the above codes we were able to build a python module to classify a given dataset using all the classifiers mentioned above and give a comparison result, to let us

know which classifier works best from different set of data. The results have been discussed in next chapter.

3.4 Improved Data cleaning for sentiment analysis.

Although we will discuss the results in the next chapter, but one think that we want to make clear before starting the discussion for the current heading is that, using the previous method (discussed in section 3.2) we would get at max 78 – 79 % accuracy. Now if we wish to enhance it we need to perform data cleaning to get rid of redundant data before classification itself. This is because the quality of data and its usefulness directly affects the ability of learning of the Machine Learning model. The various data cleaning methods we have applied are as follows:

- Converting the complete data into standard encoding format, UTF-8 (as it is widely accepted format).

```
Tweet = original_tweet. encode('ascii','ignore').decode("utf8").
```

- Removal of URLs, hyperlinks using Regular expression.
- Removal of characters like #, @, "RT"s etc. using Regular expression.
- Removal of punctuations (.,?, ! etc.) and digits (1, 12, 122, etc.).
- Replacing emoji with their corresponding words like "happy", "sad", etc.
- Removal of symbols and pictographs, transport and map symbols, flags using Regular Expressions (this are now new trends of social media)
- Splitting the attached words like GoingForVacation to Going For vacation, using Regular expressions.
- Removing the slangs used in social media with their corresponding meaning like bff to best friend forever. This has been accomplished using a dictionary lookup.
- Standardizing words like happppppy to happy, to their proper format using Regular expression has been accomplished.
- Removal of commonly occurring words i.e., stop words using nltk corpus for stop words
- Instead of using stemming, using lemmatization (Word- Net lemmatizer).

Here we have intentionally used lemmatization instead of stemming because lemmatization reduces the inflected words properly to their respective root words that belongs to the language, which stemming sometimes fails. In lemmatization root word is called Lemma. The ability of a lemmatizer comes from its use of POS tags, which it uses to understand the content first before converting. Python NLTK provides Word Net lemmatizer which uses the

Word Net Database to lookup lemmas of words. Moreover we have also implemented spell checker which uses Levenshtein Distance algorithm for correcting spellings.

The results of data cleaning will be discussed shortly, but it's time to discuss one of the most important step for sentiment classification using machine learning approach, i.e., feature extraction and feature selection.

3.5 Feature Extraction and Feature Selection.

Features help classifiers to train themselves and then classify the new instances presented before them. To implement Machine learning algorithms on text mining we need to convert texts into vectors of numbers. This conversion is termed as feature extraction and bag-of-words model of text is one of the most commonly used feature extraction method. Now Emoticons can be used to predict a particular emotion like happy or sad and cannot be used to train a classifier and it can be highly misleading. Hence, text words are generally extracted as features. Feature selection actually reduces the dimension of our data that makes the training faster and also improves the accuracy by removing irrelevant features. It also helps to avoid over fitting. So we have applied three methods for feature selection in order to reduce the dimension and enhance classifier's performance.

- A slight modification of bag-of-words (BOWs) approach. We first of all built bag-of-words for all the tweets (positive and negative tweets combined) and chose top n(for our case n = 5000) most frequently occurring in the whole corpus. We then used it on our whole corpus for training of our classifier. Experimental results have shown success, which we have discussed in next chapter.

- The second method involves an improved version of bag-of-word approach. Here first of all we separated the dataset of positive and negative tweets. We built separate bag-of-words for them separately and chose top n (for our case n = 2500) most frequently occurring words from each. We then combined them and used it on our whole corpus for training of our classifier. Experimental results have shown success, which we have discussed in next chapter.

- In our last approach, for feature set instead of using frequency of n-grams, we have implemented TF-IDF algorithm, which assigns tf-idf scores to n-grams. TF or term-frequency helps to increase the weight of an n-gram that appears more frequently within a tweet. It is defined as

  $$tf(n, tw) = \log(F(n, tw))$$

  Where, F(n, tw) is number of occurrences of n-gram 'n' in tweet 'tw'.

  On the other hand, IDF or Inverse Document Frequency helps to diminish the weight of an n-gram that appears in all the tweets of the corpus and similarly increases the weight of an n-gram that appears in rare tweets across within the corpus. It is

calculated as follows

$$idf(n, D) = \log(\frac{N}{N_{n \in tw}})$$

Where, 'N' is the total number of tweets in the corpus 'D' and Nt ∈ tw is the number of tweets in which the n-gram 'n' is present.
Finally tfidf for each n-gram is calculated as follows:

$$tfidf(n, tw, D) = tf(n, tw) \times idf(n, D)$$

As discussed earlier, we have used the supervised machine learning approach for classifying the sentiment of tweets into positive and negative. We experimented on various machine learning algorithms in order to come up with the best among them. Then with the help of data pre-processing (various data cleaning methods) and the selected machine learning classifier we have successfully performed various experiments. We got some greater enhancements due to the incorporation of our feature selection methods. With tf-idf, we have also implemented bigrams (along with unigrams) for experimentation.
So, based upon all these implementations we obtained some results which have discussed in the next chapter.

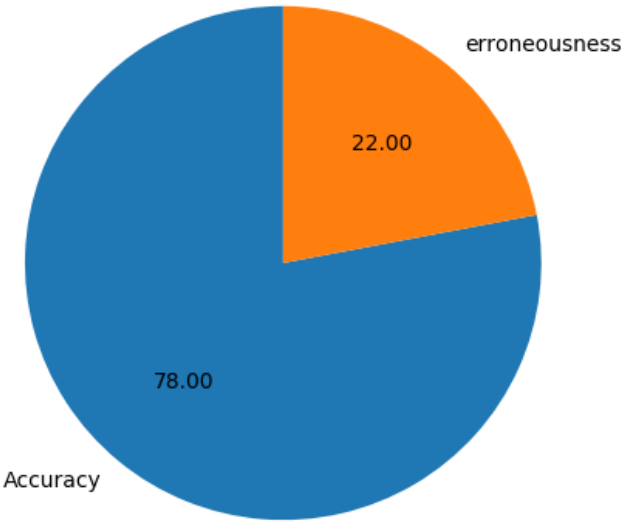
Chapter 4

Results and Discussion

In our previous chapter we have discussed the various experiments that we did in order to enhance the accuracy level of sentiment analysis. In this chapter we will look into the results of various experiments and will discuss the benefits of one method over other.

4.1 Results for use of NLTK to perform sentiment analysis.

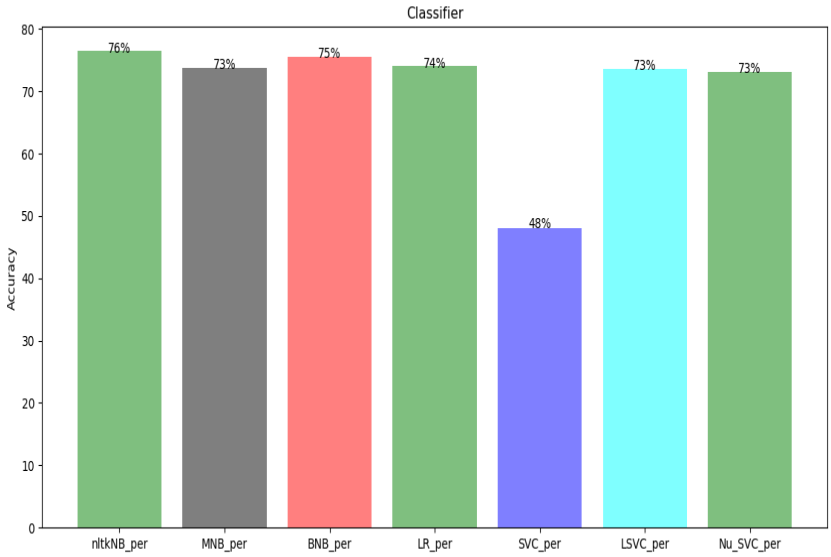
We started our journey of building the enhanced sentiment classifier module with the implementation of some of the NLTK tools and used the Naïve Bayes Classifier alone to classify the 500 unknown test data, based on its training on 1500 movie reviews (a dataset taken from the NLTK movie reviews), the result was as follows:



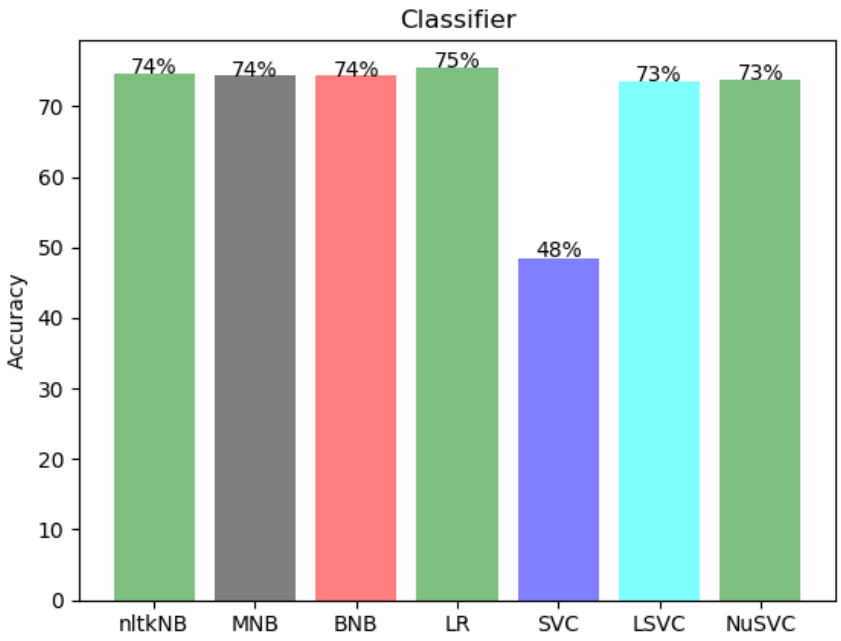
Graph 4.1: Naïve Bayes classifier on Movie reviews.

Graph 4.1 predicts the accuracy for the classification when Naïve Bayes classifier was used with some basic data pre-processing steps (like stop word removal, tokenization etc.), as mentioned in last chapter. The accuracy obtained is although not that much satisfactory (being 78%).

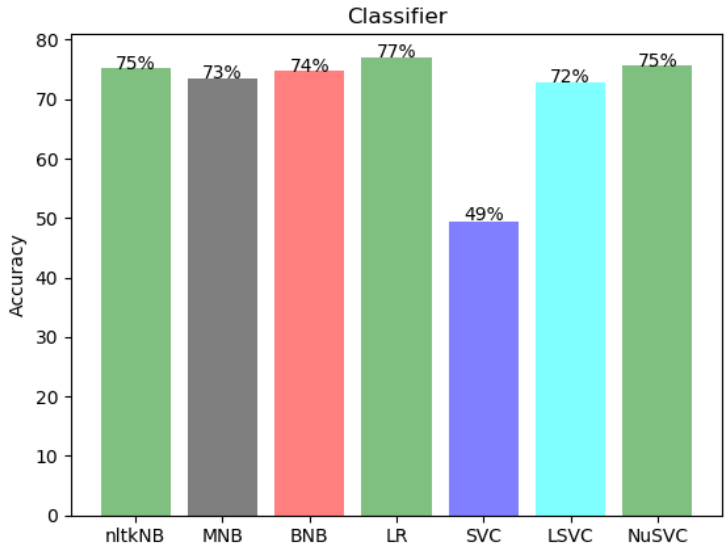
Then we thought of a new idea, we used all the classifiers together, named NB, MultinomialNB, BernoulliNB, LogisticRegression various model of SVM named SVC, LinearSVC, NuSVC in order to find out the best classifiers among them for the purpose of sentiment classification. But this we trained them using the twitter data that has been downloaded from internet. The training and testing conditions were kept same for all the classifiers upon which we got the following outputs:



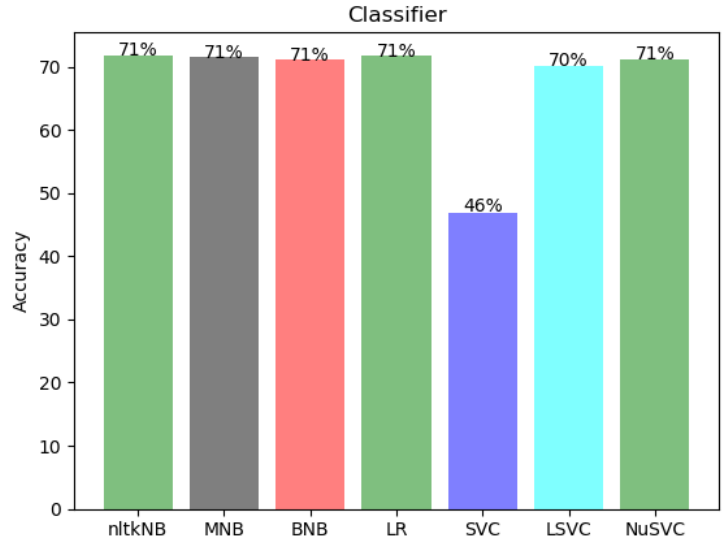
Graph 4.2: Testing 7 classifiers together (using k-folds validation 1)



Graph 4.3: Testing 7 classifiers together (using k-folds validation 3)



Graph 4.4: Testing 7 classifiers together (using k-folds validation 7)



Graph 4.5: Testing 7 classifiers together (using k-folds validation 10)

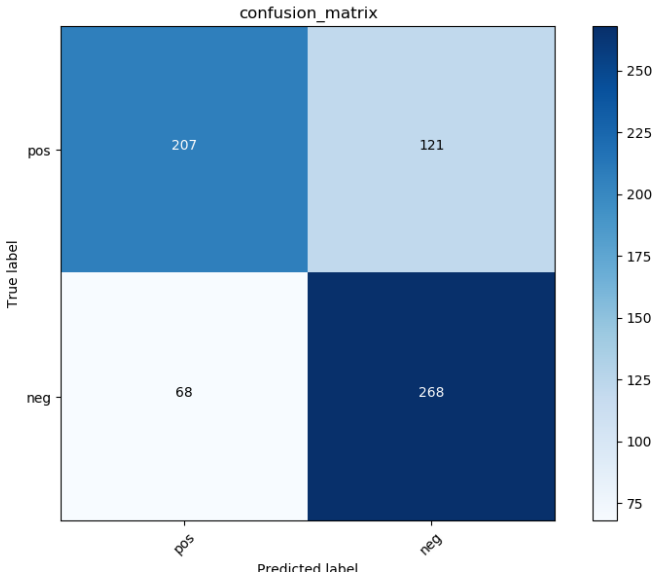
From the graphs we thus concluded that all the Naïve Bayes based classifiers works much better than all other classifiers for the purpose of sentiment analysis (although SVC variants are also good). Graph 4.2, 4.3 4.4 and 4.5 collectively displays the result of various classifiers that we have obtained using k-fold cross-validation; out of 10 results we provided the best 4 to get a better comparative analysis.

Although we can see that the Naïve Bayes classifiers are ahead of other classifiers, but the best accuracy that these algorithms can achieve is somewhat less than 80%. Hence, we got

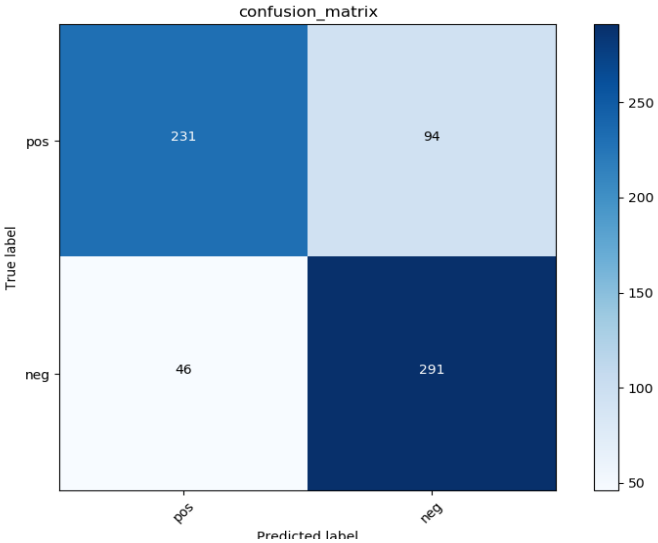
the challenge to device some technique using which we can enhance the accuracy further. Finally after a lot of research, we found that the better the input is to a machine learning model, the more will be the accuracy of the output. Hence our next experiments were with various data cleaning methods that we have already mentioned in our previous chapter.

4.2 Results for Improved Data cleaning for sentiment analysis.

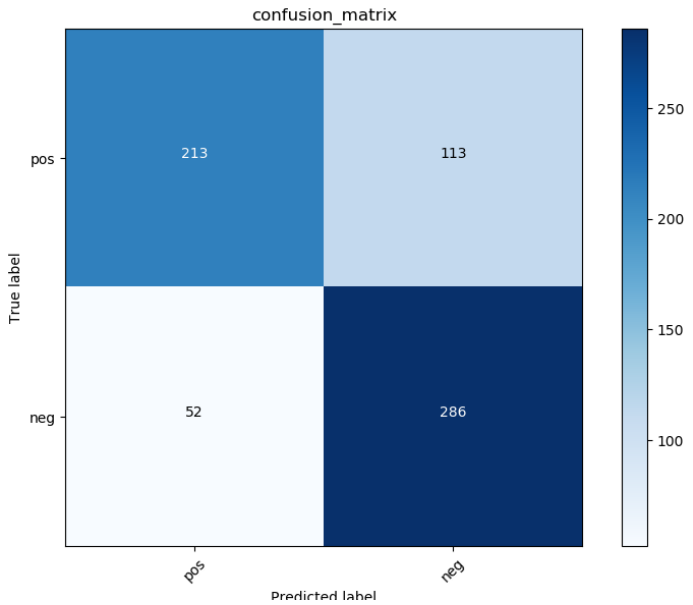
Here not going into the details of the data cleaning methods (as already been discussed in the previous chapter), let us see the comparative results that we obtained from with and without data cleaning classification using Naïve Bayes as our classifer.



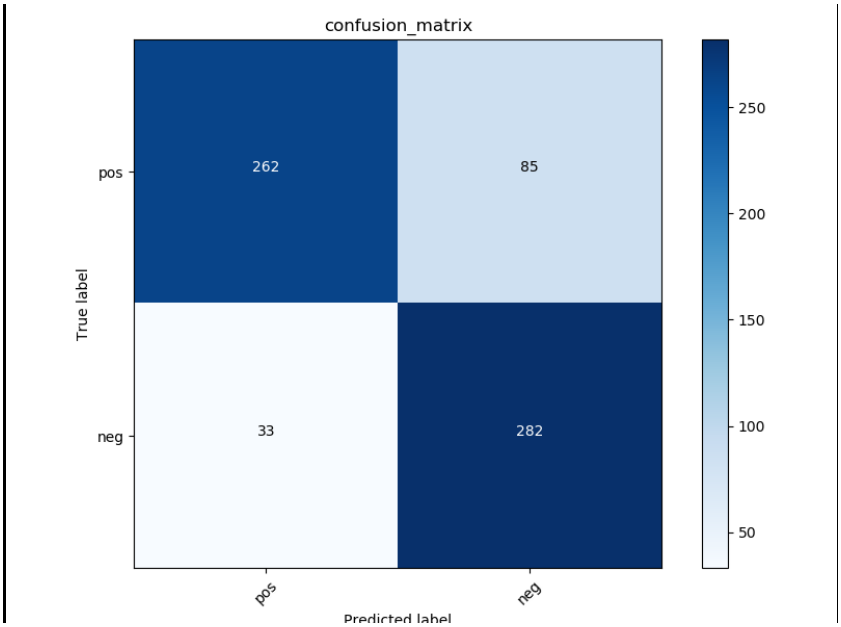
Graph 4.6a: Without Data Cleaning part1(Accuracy = 71.53)



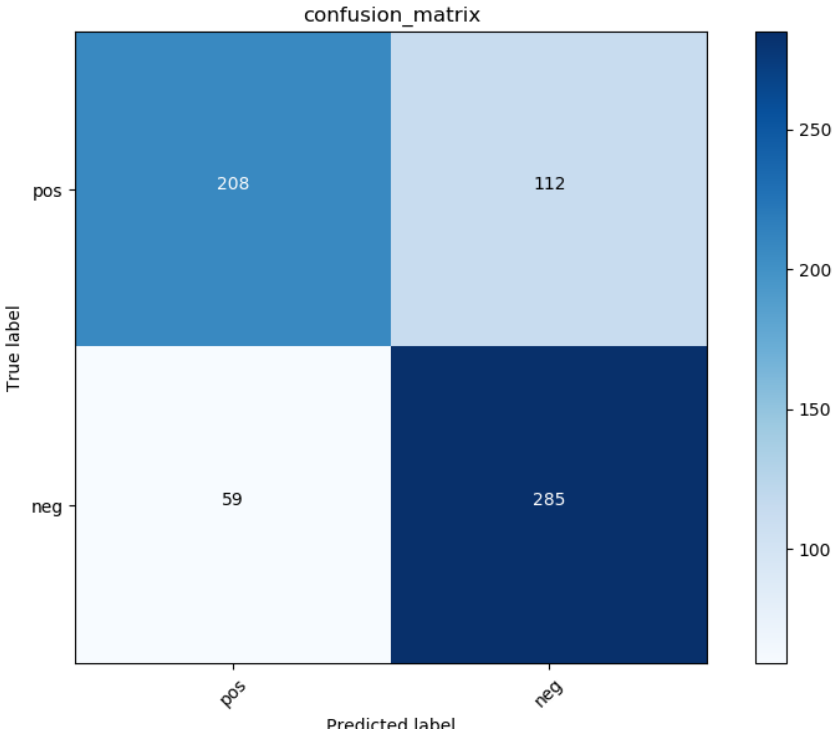
Graph 4.6b: With Data Cleaning part1(Accuracy = 80.66)



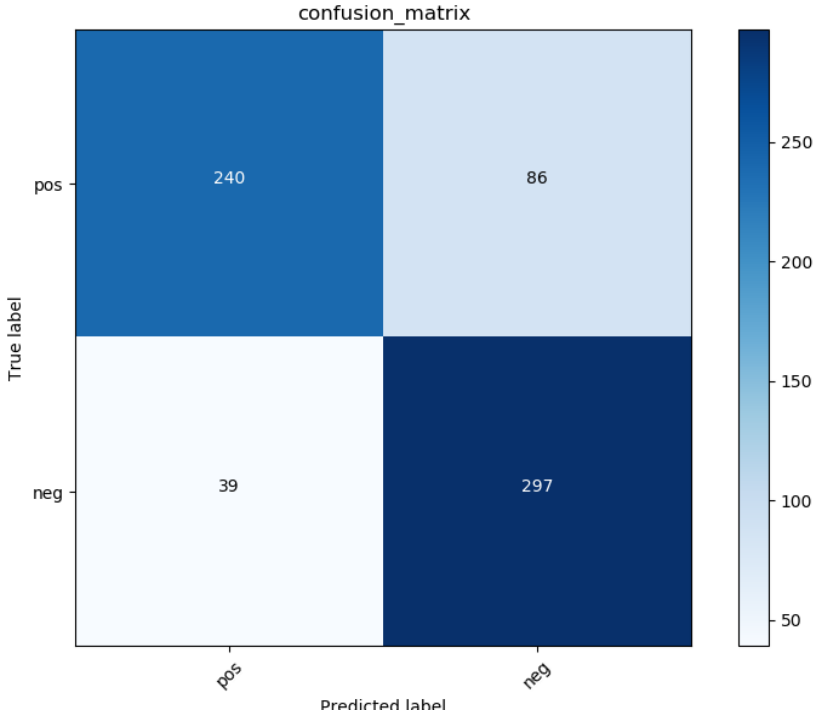
Graph 4.7a: Without Data Cleaning part2(Accuracy = 75.15)



Graph 4.7b: With Data Cleaning part2(Accuracy = 82.17)

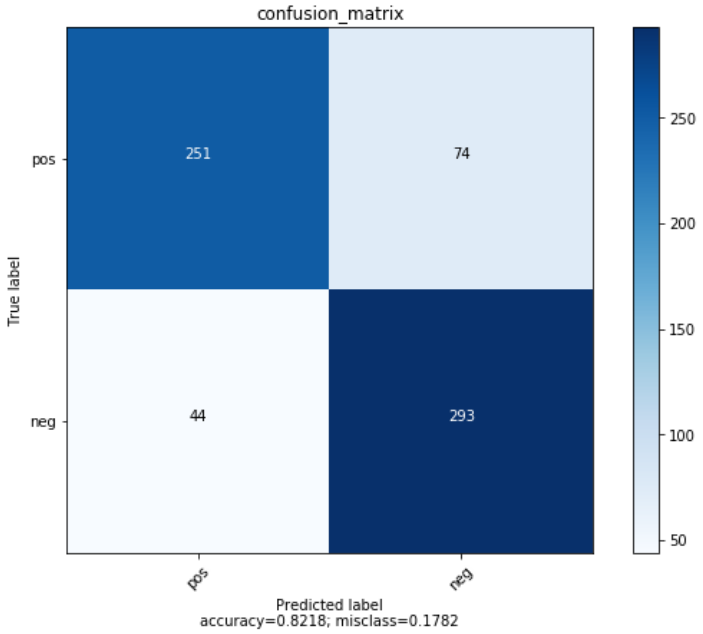


Graph 4.8a: With Data Cleaning part8(Accuracy = 81.11)



Graph 4.8b: Without Data Cleaning part8(Accuracy = 74.24)

So, from the above graphs i.e., Graph 4.6a and 4.6b, Graph 4.7a and 4.7b, Graph 4.8a and Graph4.8b, where we have shown the confusion matrices along with respective accuracies, it is clear that we have successfully achieved a hike in accuracy due to the incorporation of data cleaning processes. Thereafter instead of stemming introducing lemmatizing also gave us a positive result which has been depicted in Graph 4.9 (the reason have been already discussed in previous chapter).

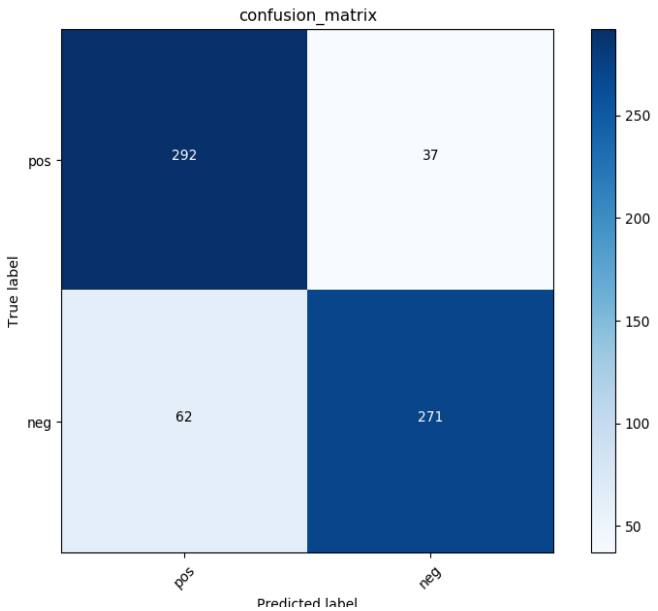


Graph 4.9: With data cleaning and Lemmatizing, Accuracy = 82.17.

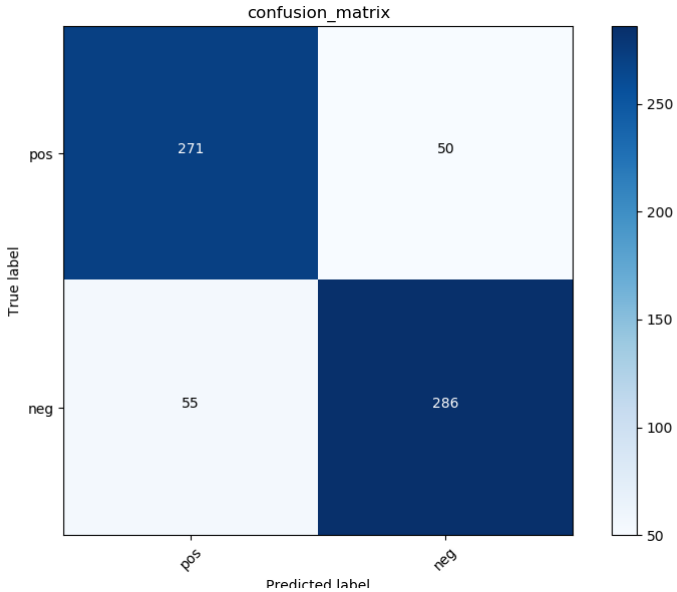
So after playing around with all suitable data cleaning techniques, we although reached up to a good accuracy level but still we wanted to explore alternatives. Now the only aspect that was known to us was that of feature extraction and feature selection. As discussed earlier, BOW is what we have chosen as our feature extraction technique but all the above graphs have been obtained using our first feature selection procedure i.e., choosing top n(for our case n = 5000) most frequently occurring in the whole corpus. Now let us see some more delightful outputs for the other two feature selection methods.

4.3 Results for other Feature extraction and Feature Selection methods.

On implementing the second method for feature selection that being selecting the top n/2 (for our case n = 5000) most frequently occurring words from each positive and negative tweet corpus and then merging them to get n feature words, we obtained the following graphs:

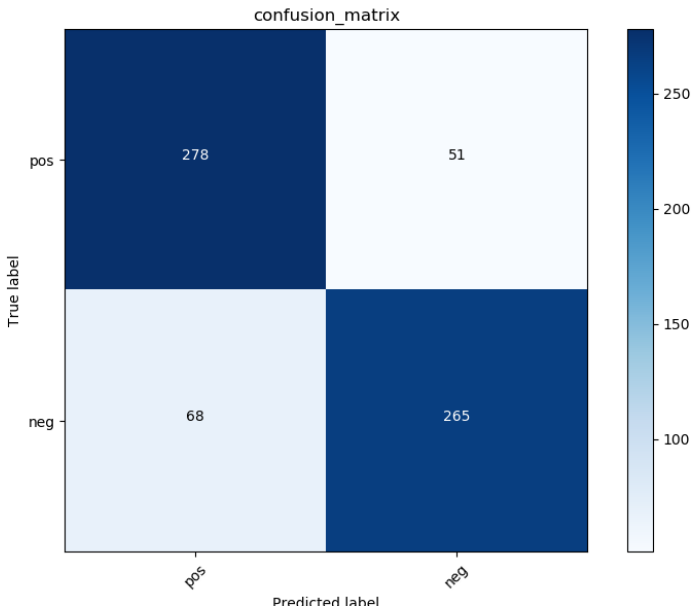


Graph 4.10: FS high frequent positive and negative tweet n-grams, Acc = 85.04.

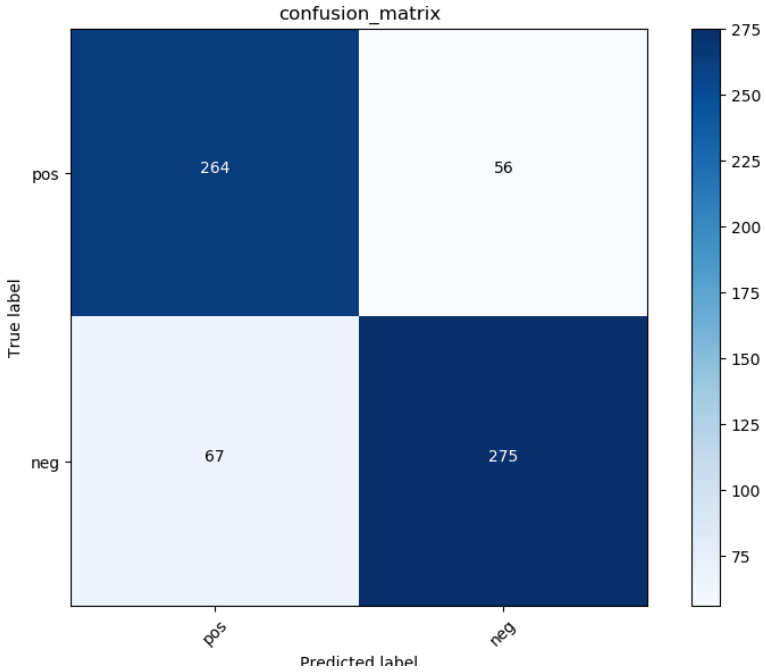


Graph 4.11: FS high frequent positive and negative tweet n-grams, Acc = 84.13.

So, from the above two graphs, Graph 4.10 and 4.11, it is pretty much clear that, this method of feature selection really was very helpful in increasing the accuracy rate for sentiment classification. But we didn't stopped here, now we instead of using frequency of words for feature extraction we chose their respective tf-idf values (calculation being discussed in previous chapter). So let us also discuss the results obtained using tf-idf as feature extraction method shown in Graph 4.12 and 4.13.



Graph 4.12: TF-IDF 1, Acc = 82.02.



Graph 4.13: TF-IDF 1, Acc = 81.41.

Thus based on all the above result we somewhat can conclude that the Feature selection method wherein we chose n/2 high frequent words from positive tweet corpus and n/2 high frequent words from negative tweet corpus gave us the best results. So, we can say that we have somewhat reached 85% accuracy mark for sentiment classification of tweets.

Chapter 5

Conclusion and Future Scope

In our previous chapters we have discussed all our experiments that we did in order to enhance the accuracy level of sentiment analysis and their corresponding results. In this chapter let us conclude our work and discuss some of the future scopes possible for this work.

4.4 Conclusion

After opting machine learning approach for sentiment analysis and going through all our present work and their results we came to the conclusion that among all the present classifiers, Naïve Bayes works better than others for sentiment analysis. We have used Bag of words as our feature words, where we chose the top most 5000 (n) words from the whole corpus, positive and negative together as our features. Later we took 2500 (n/2) high frequent words from positive and negative corpus respectively and merged them to obtain our results. We have also used tf-idf algorithm instead of frequency to weight our n-grams i.e., to select our feature words. We have used various data cleaning strategies. All our implementations aided in increasing the accuracy of sentiment classifier model. So based upon our results, we can conclude that the use of required data cleaning and proper selection of feature extraction and selection methods can improve the results for sentiment analysis.

4.5 Future Scope

After going through various blogs and research papers, we have come to know that bi-grams and tri-grams can help in giving us more accurate prediction as compared to unigrams. Although we have applied bigrams along with unigrams for tfidf, but now for our future work, we can try to use bi-grams and tri-grams instead of unigrams with the frequency approaches and can also try some more possibilities so as to enhance accuracy.

Moreover, we should try to use deep learning packages like word2Vec, GLoVE, which has more potential for enhancing accuracy, as they also takes care of the meaning and context of n-grams i.e., they preserves the semantics of words while performing classification. This technique of course has more potential for giving better results as because it can preserve the relation among words.

References

Papers

[1] Bo Pang and Lillian Lee, “A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts”, Depart- ment of Computer Science Cornell University Ithaca, NY 14853-7501

[2] Alexander Pak, Patrick Paroubek, “Twitter as a Corpus for Sentiment Analysis and Opinion Mining”, Universit´e de Paris-Sud, Laboratoire LIMSI-CNRS, Bˆatiment 508, F-91405 Orsay Cedex, France, 2009.

[3] Mohamed M. Mostafa, “More than words: Social networks’ text min- ing for consumer brand sentiments” Instituto Universita´rio de Lisboa, Business Research Unit, Avenida das Forc¸as Armadas, Lisbon, Portugal, 2013.

[4] Nan Li, Desheng Dash, “Using text mining and sentiment analysis for online forums hotspot detection and forecast”,Department of Computer Science, University of California, Santa Barbara, USA Reykjav´ık Uni- versity, Iceland RiskLab, University of Toronto, Canada, 2009

[5] Ronen Feldman, “Techniques and Applications for Sentiment Analysis” doi:10.1145/2436256.2436274, 2013

[6] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani, “SENTI- WORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining”, Istituto di Scienza e Tecnologie dell’Informazione Consiglio Nazionale delle Ricerche Via Giuseppe Moruzzi 1, 56124 Pisa, Italy.

[8] Cataldo Musto, Giovanni Semeraro, Marco Polignano, “A comparison of Lexicon-based approaches for Sentiment Analysis of microblog posts”,Department of Computer Science University of Bari Aldo Moro, Italy, 2014

[9] Walaa Medhat , Ahmed Hassan b, Hoda Korashy b, ‘Sentiment analysis algorithms and applications: A survey”, School of Electronic Engineering, Canadian International College, Cairo Campus of CBU, Egypt b Ain Shams University, Faculty of Engineering, Computers & Systems Department, Egypt, May, 2014

[11] Vallikannu Ramanathan, T.Meyyappan, “Twitter Text Mining for Sen- timent Analysis on People’s Feedback about Oman Tourism”,Research Scholar Professor Alagappa University Alagappa University Karaikudi, Tamilnadu Karaikudi, Tamilnadu India India, 2019

[12] Radhi D. Desai, “Sentiment Analysis of Twitter Data”, Department of Computer Engineering, Sardar Vallabhbhai Patel Institute of Technology, Vasad, Gujarat- 388 306, INDIA, 2018.

[13] Bin Lin, Fiorella Zampetti, Rocco Oliveto, Massimiliano Di Penta, Michele Lanza, and Gabriele Bavota, "Two Datasets for Sentiment Analysis in Software Engineering", Universit'a della Svizzera italiana (USI), Switzerland — University of Sannio, Italy — ‡University of Molise, Italy, 2018.

[14] Prerna Mishra, Dr. Ranjana Rajnish, Dr.Pankaj Kumar, "Sentiment Anal- ysis of Twitter Data:Case Study on Digital India",Research Scholar-IT Amity University Lucknow, India, Assistant. Professor Amity University Lucknow, India, Assistant Professor SRMGPC Lucknow, India , 2016

[15] M.Trupthi, Suresh Pabboju and G.Narasimha, "SENTIMENT ANAL- YSIS ON TWITTER USING STREAMING API", Computer Science Department, JNTUH, Telangana State, India. Suresh Pabboju2, Infor- mation Technology Department, CBIT, Hyderabad, Telangana State, India. G.Narasimha3, Computer ScienceDepartment, JNTUH, Jagital, Telangana State, India, 2017

[16] Esha Tyagi, Dr. Arvind Kumar Sharma, "An Intelligent Framework for Sentiment Analysis of Text and Emotions – A Review", Career Point University Kota, Rajasthan-India, 2017.

[17] Rasika Wagh and Payal Punde, "Survey on Sentiment Analysis using Twitter Dataset", (Mtech): Department of Computer Science and IT, Dr. BAMU Aurangabad, India, (Mtech): Department of Computer Science and IT, Dr. BAMU Aurangabad, India, 2018.

[18] Victoria Ikoro1, Maria Sharmina2, Khaleel Malik3 and Riza Batista- Navarro, "Analyzing Sentiments Expressed on Twitter by UK Energy Company Consumers", School of Computer Science, Alliance Manch- ester Business School, Tyndall Centre for Climate Change Research University of Manchester, Manchester, United Kingdom, 2018

Links

1. https://www.datasciencecentral.com/profiles/blogs/text-mining-and-sentiment-analyses-a-primer
2. http://blog.aylien.com/first-text-mining-project-python-3-steps/
3. https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
4. https://www.ibmbigdatahub.com/blog/introduction-text-mining
5. https://elitedatascience.com/python-nlp-libraries
6. https://developer.twitter.com/apps
7. https://pythonprogramming.net/text-classification-nltk-tutorial/
8. https://www.nltk.org/
9. https://hub.packtpub.com/clean-social-media-data-analysis-python/
10. https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
11. https://www.ibmbigdatahub.com/blog/introduction-text-mining
12. https://elitedatascience.com/python-nlp-libraries