

# 1. Introduction

Most of the time reviews on movies carry sentiment which indicates whether review is positive or negative. The goal of this project report is to explain how a system is developed to predict the sentiments of reviews using basic algorithms and compare the results.

The human language is complex. Teaching a machine to analyze the various grammatical nuances, cultural variations, slang and misspellings that occur in online mentions is a difficult process. Teaching a machine to understand how context can affect tone is even more difficult.

Humans are fairly intuitive when it comes to interpreting the tone of a piece of writing. Consider the following sentence: “My flight’s been delayed. Brilliant!”

Most humans would be able to quickly interpret that the person was being sarcastic. We know that for most people having a delayed flight is not a good experience (unless there’s a free bar as recompense involved). By applying this contextual understanding to the sentence, we can easily identify the sentiment as negative.

Without contextual understanding, a machine looking at the sentence above might see the word “brilliant” and categorize it as positive.

## **1.1 Sentiment Analysis**

Sentiment analysis refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. It is also known as opinion mining or emotion AI.

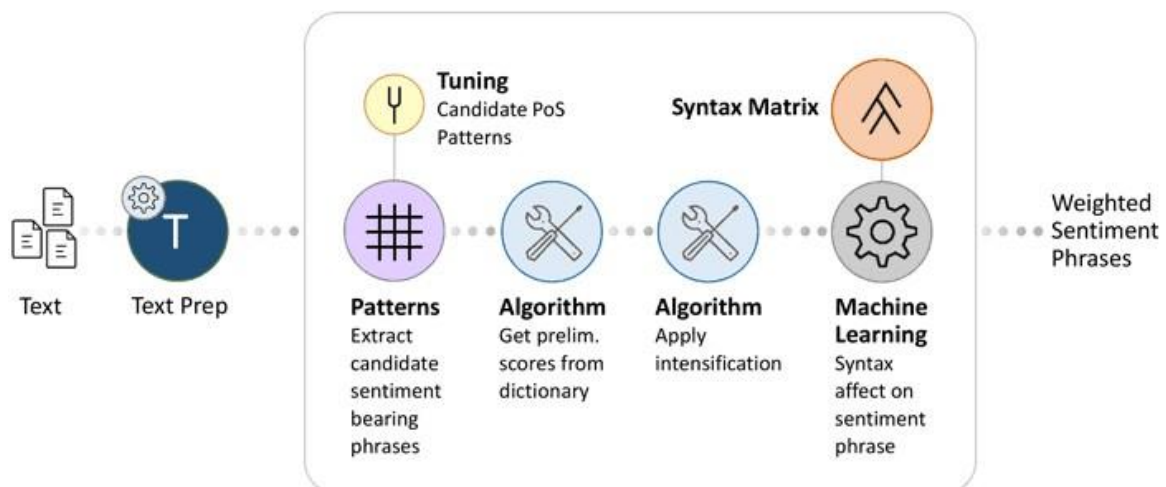
According to [2] an opinion “is simply a positive or negative sentiment, view, attitude, emotion, or appraisal about an entity or an aspect of the entity” from an opinion holder at a specific time. The entity can be a product/service, event, person, organization, or topic consisting of aspects (features/attributes) that represents both components and attributes of the entity. The term sentiment analysis first appeared in [3] which explored the essential issues in sentiment analysis to identify how sentiments are expressed in texts and whether the expressions indicate positive (favorable) or negative (unfavorable) opinions toward the subject.

In order to calculate the sentiment score of the review, each piece of text can be examined separately or in combination with others. In this manner, after calculating the sentiment scores of all the pieces of text in the review some aggregation technique is used to calculate the overall sentiment of the review. One of the simplest ways of combining the total score of the review is summing all the scores of all the pieces in that review. Certainly, it is our choice to select how big those pieces will be.

We can choose every word, n subsequent words, sentence, and/or whole review to represent a feature. However, due to the fact that there is a little chance that there will be repeating reviews, it is more reasonable to have smaller pieces of text as representations of features.

## 1.2 General Approach to Sentiment Analysis

The sentiment analysis is a complex process that involves 5 different steps to analyze sentiment data. These steps are explained below.



**Figure 1.2.1 General Approach to Sentiment Analysis**

- Data collection: The first step of sentiment analysis consists of collecting data from user generated content contained in blogs, forums, social networks. These data are disorganized, expressed in different ways by using different vocabularies, slangs, context of writing etc. Manual analysis is almost impossible due to the

voluminous content. Therefore, text analytics and natural language processing are used to extract and classify.

- Text preparation: It consists in cleaning the extracted data before analysis. Non-textual contents and contents that are irrelevant for the analysis are identified and eliminated.
- Sentiment detection: The extracted sentences of the reviews and opinions are examined. Sentences with subjective expressions (opinions, beliefs and views) are retained and sentences with objective communication (facts, factual information) are discarded.
- Sentiment classification: In this step, subjective sentences are classified in positive, negative, good, bad; like, dislike, but classification can be made by using multiple points.
- Presentation of output: The main objective of sentiment analysis is to convert unstructured text into meaningful information. When the analysis is finished, the text results are displayed on graphs like pie chart, bar chart and line graphs. Also, time can be analyzed and can be graphically displayed constructing a sentiment time line with the chosen value (frequency, percentages, and averages) over time.

**Table 1.2.1 Sentiment classification approaches**

<b>SENTIMENT CLASSIFICATION APPROACHES</b>		<b>FEATURES/TECNIQUES</b>	<b>ADVANTAGES AND LIMITATIONS</b>
<b>Machine learning</b>	Bayesian Networks Naive Bayes Classification Maximum Entropy Neural Networks Support Vector Machine	Part of speech information Negations Opinion words and phrases Term presence and frequency	ADVANTAGES the ability to adapt and create trained models for specific purposes and contexts LIMITATIONS the low applicability to new data because it is necessary the availability of labeled data that could be costly or even prohibitive
<b>Lexicon based</b>	Dictionary based approach Novel Machine Learning Approach Corpus based approach Ensemble Approaches	Manual construction, Corpus-based Dictionary based	ADVANTAGES wider term coverage LIMITATIONS finite number of words in the lexicons and the assignation of a fixed sentiment orientation and score to words
<b>Hybrid</b>	Machine learning Lexicon based	Sentiment lexicon constructed using public resources for initial sentiment detection Sentiment words as features in machine learning method	ADVANTAGES lexicon/learning symbiosis, the detection and measurement of sentiment at the concept level and the lesser sensitivity to changes in topic domain LIMITATIONS noisy reviews

The sentiment classification approaches can be classified three types. The machine learning approach is used for predicting the polarity of sentiments based on trained as well as test data sets. While the lexicon based approach does not need any prior training in order to mine the data. It uses a predefined list of words, where each word is associated with a specific sentiment. Finally, in the hybrid approach, the combination of both the machine learning and the lexicon based approaches has the potential to improve the sentiment classification performance.

### **1.3 Applications**

The uses of sentiment analysis are listed below:

- i) Inform and make operational improvements or capital expenditures. With the Sentiment Analysis report, a hotelier could look at a recent time period and identify topics that were mentioned most frequently with a negative sentiment attached. In the topic cloud below, the topics that came up most frequently are “room odor,” “noise,” and “bathroom condition.” With this information, a hotel would know to evaluate these areas when it comes time to make decisions about property improvements. The applications of sentiment analysis in business cannot

overlooked. Sentiment analysis in business can prove a breakthrough for the complete brand revitalization.

- ii) More and more we're seeing it used in social media monitoring and Voice of the Customer (VoC) to track customer reviews, survey responses, competitors, etc.
- iii) To forecast market movement based on news, blogs and social media sentiment.
- iv) For a recommender system, sentiment analysis has been proven to be a valuable technique. A recommender system aims to predict the preference to an item of a target user. Mainstream recommender systems work on explicit data set.

## **1.4 Approaches and Dataset**

Two approaches are used in the system created for sentiment analysis.

They are as follows:

- i) Most common words
- ii) tf-idf

One approach is to select the terms, specifically adjectives as result of POS tagging, that are most common in class. It selects all the terms that have been appeared a considerable number of times and thus having a high probability of classifying given review accurately.

Sometimes the most repetitive words are not helpful in classifying the sentiment accurately and certain weights need to be assigned to every term based on relative importance of features. These features are individual words or word n-grams and their frequency counts. It uses term frequency weights to indicate the relative importance of features.

TF-IDF is not a single method, but a class of techniques where similarity between queries and documents is measured via the sum of term frequency-like numbers (TFs) multiplied by terms' importance. The term importance is frequently expressed via the IDF (the inverse document frequency, actually is the logarithm of IDF that is used in practice).

A query term is not a good discriminator if it occurs in many documents. We should give it less weight than one occurring in few documents. For examples, when querying “information retrieval”, it is unlikely that documents containing “information” might be more relevant than documents containing “retrieval”. In 1972, Spärck Jones introduced a measure of term specificity (discriminative power) called Inverse Document Frequency (IDF).

The dataset used in the proposed system is Cornell Dataset. This corpus is the collection of movie-review documents labeled with respect to their overall sentiment polarity (positive or negative) or subjective rating (e.g., "two and a half stars") and sentences labeled with respect to their subjectivity status (subjective or objective) or polarity.



## **2. Proposed System**

### **2.1 Algorithms Used**

#### **2.1.1 Naïve Bayes**

The Bayesian Classification represents a supervised learning method as well as a statistical method for classification. It 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. It can solve diagnostic and predictive problems.

Bayesian classification provides practical learning algorithms and prior knowledge and observed data can be combined. Bayesian Classification provides a useful perspective for understanding and evaluating many learning algorithms. It calculates explicit probabilities for hypothesis and it is robust to noise in input data.

It is based on the Bayesian theorem. It is particularly suited when the dimensionality of the inputs is high. Parameter estimation for naive Bayes models uses the method of maximum likelihood. It requires a small amount of training data to estimate the parameters.

Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where  $A$  and  $B$  are events and  $P(B) \neq 0$ .

- $P(A)$  and  $P(B)$  are the probabilities of observing  $A$  and  $B$  without regard to each other.
- $P(A|B)$ , a conditional probability, is the probability of observing event  $A$  given that  $B$  is true.
- $P(B|A)$  is the probability of observing event  $B$  given that  $A$  is true.

We can also say that:

$$Posterior = \frac{Likelihood \times Prior}{Evidence}$$

- There are several Naive Bayes Variations. Here we will use 2 of them: The Multinomial Naive Bayes and the Bernoulli Naive Bayes. Multinomial Naive Bayes is used when the multiple occurrences of the words matter a lot in the classification problem. Such an example is when we try to perform Topic Classification. The Bernoulli Naive Bayes can be used when in our problem the absence of a particular word matters. For example Bernoulli is commonly used in Spam or Adult Content Detection with very good results. It is useful when feature vectors follows multinomial distribution. It is specifically used for discrete counts

### **2.1.1.1 Multinomial Naive Bayes**

This variation, as described by Manning et al (2008) [9], estimates the conditional probability of a particular word/term/token given a class as the relative frequency of term  $t$  in documents belonging to class  $c$ :

$$P(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

Thus, this variation takes into account the number of occurrences of term  $t$  in training documents from class  $c$ , including multiple occurrences.

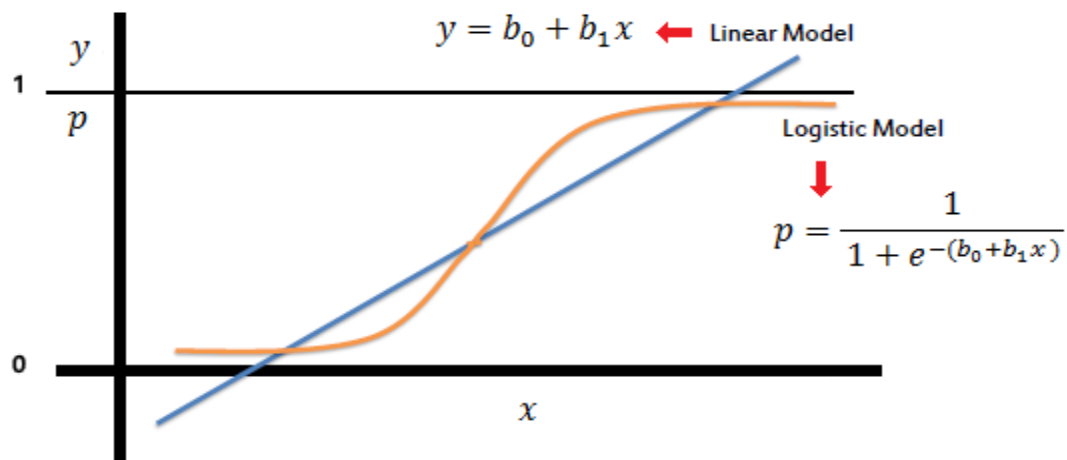
### **2.1.1.2 Bernoulli Naive Bayes**

The Bernoulli variation, as described by Manning et al (2008) [10], generates a Boolean indicator about each term of the vocabulary equal to 1 if the term belongs to the examining document and 0 if it does not. It is useful when feature vectors are binary. The model of this variation is significantly different from Multinomial not only because it does not take into consideration the number of occurrences of each word, but also because it takes into account the non-occurring terms within the document. While in Multinomial model, the non-occurring terms are completely ignored. The Bernoulli model has the same time complexity as the multinomial model. Like MultinomialNB, this classifier is suitable for discrete data.

### 2.1.2 Logistic Regression

Logistic regression belongs to the family of classifiers known as the exponential or log-linear classifiers. Like naive Bayes, it log-linear classifier works by extracting some set of weighted features from the input, taking logs, and combining them linearly (meaning that each feature is multiplied by a weight and then added up). Technically, logistic regression refers to a classifier that classifies an observation into one of two classes, and multinomial logistic regression is used when classifying into more than two classes.

The most important difference between naive Bayes and logistic regression is that logistic regression is a discriminative classifier while naive Bayes is a generative classifier.



**Fig 2.1.2.1 Difference between a linear and logistic model**

### **2.1.3 Linear Support Vector Machine**

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a new data point will be in. In the case of support vector machines, a data point is viewed as a  $p$ -dimensional vector (a list of  $p$  numbers), and we want to know whether we can separate such points with a  $(p-1)$ -dimensional hyperplane. This is called a linear classifier.

In above figure,  $H_1$  does not separate the classes.  $H_2$  does, but only with a small margin.  $H_3$  separates them with the maximum margin.

Linear SVM is a linearly scalable routine meaning that it creates an SVM model in a CPU time which scales linearly with the size of the training data set. It is efficient in dealing with extra-large data sets (say, several millions training data pairs).

### **2.1.4 Stochastic Gradient Descent**

Stochastic gradient descent (SGD) is a gradient descent optimization method that minimizes a given loss function. The term “stochastic” refers to the fact that the weights of the model are updated for each training

example, which is an approximation of batch gradient , in which all training examples are considered to make a single step. This way SGD is very fast to train.

### **2.1.5 AdaBoost**

It is a Machine Learning algorithm proposed by Freund and Robert Schapire who won the Gödel Prize in 2003 for their work. It can be used in conjunction with many other types of learning algorithms to improve their performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems, it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing (e.g., their error rate is smaller than 0.5 for binary classification), the final model can be proven to converge to a strong learner.

While every learning algorithm will tend to suit some problem types better than others, and will typically have many different parameters and configurations to be adjusted before achieving optimal performance on a dataset, AdaBoost (with decision trees as the weak learners) is

often referred to as the best out-of-the-box classifier. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples.

### **2.1.6 Random Forest**

Random forests operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

You can get an idea of the mechanism from the name itself-"random forests". A collection of trees is a forest, and the trees are being trained on subsets which are being selected at random, hence random forests.

Decision trees are a popular method for various machine learning tasks. Tree learning "come[s] closest to meeting the requirements for serving as an off-the-shelf procedure for data mining", say Hastie *et al.*, because it is invariant under scaling and various other transformations of feature values, is robust to inclusion of irrelevant features, and produces inspect able models. However, they are seldom accurate.

In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, i.e. have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with

the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

## **2.2 Flow of Operations**

### **2.2.1 Input and Dataset**

We use the data provided in [15], as the input to the proposed system. Here is the description of the data:

This corpus is the collection of movie-review documents labeled with respect to their overall sentiment polarity (positive or negative). The labeled data set consists of 10,662 IMDB movie reviews, specially selected for sentiment analysis. The dataset consists of 5331 positive and 5331 negative processed sentences / snippets. The above dataset is being used for training and testing of the model where in 80% of the data is used for training and the rest is used as test data.

### **2.2.2 Data Preprocessing**

Data preprocessing is a technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of



resolving such issues. Data preprocessing prepares raw data for further processing.

Online texts contain lots of noise that can cause misleading results in the sentiment classification process, such as html tags, advertisements, hyperlinks, stop words and words that bear no effect on the text orientation. Keeping those words makes the dimensionality of the problem high and hence the classification more difficult since each word in the text is treated as one dimension. Here is the hypothesis of having the data properly pre-processed: to reduce the noise in the text thereby helping improve the performance of the classifier and speed up the classification process, thus aiding in real time sentiment analysis.

#### **2.2.2.1 Tokenizing**

Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. A *token* is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing. The tokens can be used further for parsing (syntactic analysis) or text mining. Tokenization is generally considered easy relative to other tasks in text mining and also one of the uninteresting phases. However, errors made in this phase will propagate into later phases and cause problems.

The first step in majority of text processing applications is to segment text into words. In English and other European languages, word tokens are delimited by a blank space. Thus, for such languages, which are called segmented languages token boundary identification is a somewhat trivial task since the majority of tokens are bound by explicit separators like spaces and punctuation. A simple program which replaces white spaces with word boundaries and cuts off leading and trailing quotation marks, parentheses and punctuation produces an acceptable performance.

The next step is to handle abbreviations. In English and other European languages even though a period is directly attached to the previous word, it is usually a separate token which signals the termination of the sentence. However, when a period follows an abbreviation it is an integral part of this abbreviation and should be tokenized together with it. The Dr. is pleased. Now, if we ignore addressing the challenge posed by abbreviation, this line would be delimited into The is pleased. Universally accepted standards for many abbreviations and acronyms do not yet exist. The most common approach to the recognition of abbreviations is to maintain a list of already known abbreviations. Thus during tokenization a word with a trailing period can be looked up in such a list and, if it is found there, it is tokenized as a single token, else the period is tokenized as a separate token.

The third step is segmentation of hyphenated words which answers the question One or two words? Hyphenated segments can cause ambiguity for a tokenizer. Sometimes a hyphen is part of a token, i.e. self-assessment, G-45, thirty-five and sometimes it is not e.g. New Delhi-based. Tokenization of hyphenated words is generally task dependent. For instance, part-of-speech taggers usually treat hyphenated words as a single syntactic unit and therefore prefer them to be tokenized as single tokens. [14]

#### **2.2.2.2 Eliminating stop words**

One of the major forms of pre-processing is going to be filtering out useless data. In natural language processing, useless words (data), are referred to as stop words.

Some words carry more meaning than other words and some words are just plain useless, filler words. Such words, which are of little value in providing the semantics to the sentence, need to be identified and eliminated. Eliminating stop words results in effective utilization of classifier's feature space and its classification performance. It also helps in emphasizing on the words which have significant meaning and contribute to the overall sentiment of the sentence, thereby improving the accuracy of the classifier. One way to eliminate stop words is to generate a pre-compiled list of words that are likely to be stop words and then filter out all the words from the dataset which appear in the stop

list. In the proposed system, we will be using the stop list provided by NLTK that can be accessed via the NLTK corpus.

### **2.2.2.3 Stemming**

Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root.

There are several types of stemming algorithms which differ in respect to performance and accuracy and how certain stemming obstacles are overcome. Three common stemming algorithms in the context of sentiment:

- Porter stemmer
- Lancaster stemmer
- WordNet stemmer

Porter and Lancaster destroy too many sentiment distinctions. The WordNet stemmer does not have this problem nearly so severely, but it generally doesn't do enough collapsing to be worth the resources necessary to run it.

The most common algorithm for stemming English, and one that has repeatedly been shown to be empirically very effective, is *Porter's*

*algorithm.* Porter's algorithm consists of 5 phases of word reductions, applied sequentially. Within each phase there are various conventions to select rules, such as selecting the rule from each rule group that applies to the longest suffix.

The WordNet stemmer (NLTK) is high-precision. It requires word-POS pairs. Its only general issue for sentiment is that it removes comparative morphology.

#### **2.2.2.4 POS Tagging**

In corpus linguistics, part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context—i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph.

Part-of-speech tagging is harder than just having a list of words and their parts of speech, because some words can represent more than one part of speech at different times, and because some parts of speech are complex or unspoken.

NLTK module performs POS tagging effectively by labelling every word with its respective part of speech and also by tense. Following is a list of POS tags addressed by NLTK module:

**Table 2.2.2.4.1 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 1)
MD	modal could, will
NN	noun, singular 'desk'
NNS	noun plural 'desks'
NNP	proper noun, singular 'Harrison'
NNPS	proper noun, plural 'Americans'
PDT	predeterminer '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, superlative best
RP	particle give up
TO	to go 'to' the store.
UH	interjection errrrrrrm
VB	verb, base form take
VBD	verb, past tense took
VBG	verb, gerund/present participle taking
VC	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

When any sentence is passed to the module, it generates a list of tuples, where the first element in the tuple is the word, and the second is the part of speech tag. Following is the example output:

[('PRESIDENT', 'NNP'), ('GEORGE', 'NNP'), ('W.', 'NNP'), ('BUSH', 'NNP'), ('"S"', 'POS'), ('ADDRESS', 'NNP'), ('BEFORE', 'NNP'), ('A', 'NNP'), ('JOINT', 'NNP'), ('SESSION', 'NNP'), ('OF', 'NNP'), ('THE', 'NNP'), ('CONGRESS', 'NNP'), ('ON', 'NNP'), ('THE', 'NNP'), ('STATE', 'NNP'), ('OF', 'NNP'), ('THE', 'NNP'), ('UNION', 'NNP'), ('January', 'NNP'), ('31', 'CD'), ('.', '.'), ('2006', 'CD'), ('THE', 'DT'), ('PRESIDENT', 'NNP'), (':', ':'), ('Thank', 'NNP'), ('you', 'PRP'), ('all', 'DT'), ('.', '.')] [('Mr.', 'NNP')]

In the proposed system, we specifically extract the adjectives, verbs and adverbs in the data as they provide the semantics.

### **2.2.3 Handling Negation**

Negation plays an important role in polarity analysis. One of the example sentences from our corpus – “This is not a good movie” had the opposite polarity from the sentence “This is a good movie”, although the features of the original model would show that they were of the same polarity.

So in order to handle the word “good” in first and second sentences differently, we added the polarity of the word to it. The negation handling module changed the polarity of “good” to negative. It is very important to handle the negation as it will result in the change of entire sentiment of the sentence and it would result in the decline of overall accuracy of the system.

## 2.2.4 Feature Selection

Feature selection is often integrated as the first step in machine learning algorithms like SVM, Neural Networks etc. The main goal of the feature selection is to decrease the dimensionality of the feature space and thus computational cost. As a second objective, feature selection will reduce the overfitting of the learning scheme to the training data. During this process, it is also important to find a good tradeoff between the richness of features and the computational constraints involved when solving the categorization task.

The proposed system uses two approaches for selecting features. They are as follows:

**Most common words:** One approach is to select the terms, specifically adjectives as result of POS tagging, that are most common in class. It selects all the terms that have been appeared a considerable number of times and thus having a high probability of classifying given review accurately.

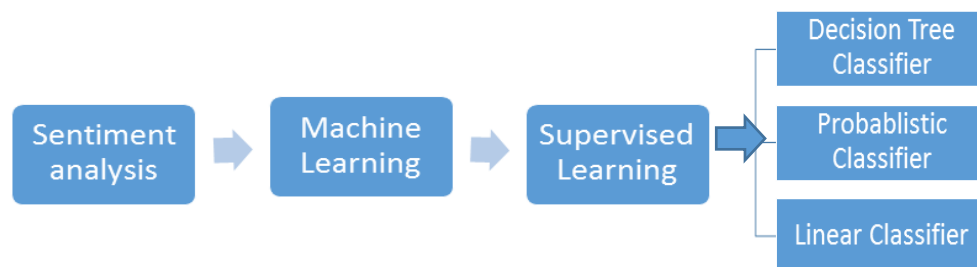
**Term presence and frequency:** Sometimes the most repetitive words are not helpful in classifying the sentiment accurately and certain weights need to be assigned to every term based on relative importance of features. These features are individual words or word n-grams and



their frequency counts. It uses term frequency weights to indicate the relative importance of features.

### **2.2.5 Classification using machine learning algorithms**

There are various classification techniques to identify the sentiment of given unstructured data. The proposed system uses Machine Learning approach for classification process. It specifically implements Supervised Learning algorithms. The proposed system uses different machine learning algorithms such as Logistic regression, Stochastic Gradient Descent, Adaptive boost algorithms and various Probabilistic Classifiers like Naïve Bayes, Multinomial Naïve Bayes, Bernoulli Naïve Bayes and Linear classifiers like Support Vector Machines are used to classify the data. The proposed system uses algorithms above and compares results of the above algorithms.



**Figure 2.2.5.1 Classification using machine learning algorithms**

## **2.3 Natural Language Toolkit(NLTK)**

### **2.3.1 Overview**

NLTK is a 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, wrappers for industrial-strength NLP libraries, and an active discussion forum. [13]

NLTK includes graphical demonstrations and sample data. It supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

### **2.3.2 Libraries and APIs used**

NLTK provides various libraries for core NLP functions like tokenizing, stemming, tagging, parsing etc. NLTK provides `nltk.tokenize` for tokenizing a sentence. Tokenization is generally considered easy relative to other tasks in text mining and also one of the uninteresting phases. However, errors made in this phase will propagate into later phases and cause problems. Here is an example of tokenizing a sentence using 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))
print(word_tokenize(EXAMPLE_TEXT))

```

NLTK provides a bunch of words that are considered to be stop and can be accessed via NLTK corpus as follows:

```

from nltk.corpus import stopwords

```

Stemming data is easy using inbuilt functions provided by NLTK module which implements Porter's Stemmer algorithm.

```

from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()

example_words=["python", "pythoner", "pythoning", "pythoned", "pyt
honly"]

for w in example_words:
    print(ps.stem(w))

```

### 3. Experiments and Results

#### 3.1 Dataset

The dataset used in the proposed system is Cornell Dataset. This corpus is the collection of movie-review documents labeled with respect to their overall sentiment polarity (positive or negative). The labeled data set consists of 10,662 IMDB movie reviews, specially selected for sentiment analysis. The dataset consists of 5331 positive and 5331 negative processed sentences / snippets.

Total number of features in the dataset: 8541

Features considered for sentiment analysis: 6000

Data	Movie Review Dataset	
	Positive reviews	Negative reviews
Number of sentences	5,331	5,331
Number of terms	116,080	116,176
Number of distinct terms	20,370	21,052
Mean number of terms per sentence	21.77	21.79
Mean number of distinct terms per sentence	25.86	23.81

Table 3.1 Corpus statistics for movie review

### **3.2 Training**

80% of the dataset i.e. 8,530 labelled sentences are used for training purpose.

Total number of labelled sentences used for training: 8530

Number of positive labelled sentences used for training: 4265

Number of negative labelled sentences used for testing: 4265

### **3.3 Testing**

20% of the dataset i.e. 2,132 labelled sentences are used for training purpose.

Total number of labelled sentences used for training: 2132

Number of positive labelled sentences used for training: 1066

Number of negative labelled sentences used for testing: 1066

### **3.4 Accuracy**

#### **Most common words approach:**

MNB\_classifier accuracy percent: 76.28398791540786

BernoulliNB\_classifier accuracy percent: 76.13293051359517

LogisticRegression\_classifier accuracy percent: 75.83081570996978

LinearSVC\_classifier accuracy percent: 72.05438066465257

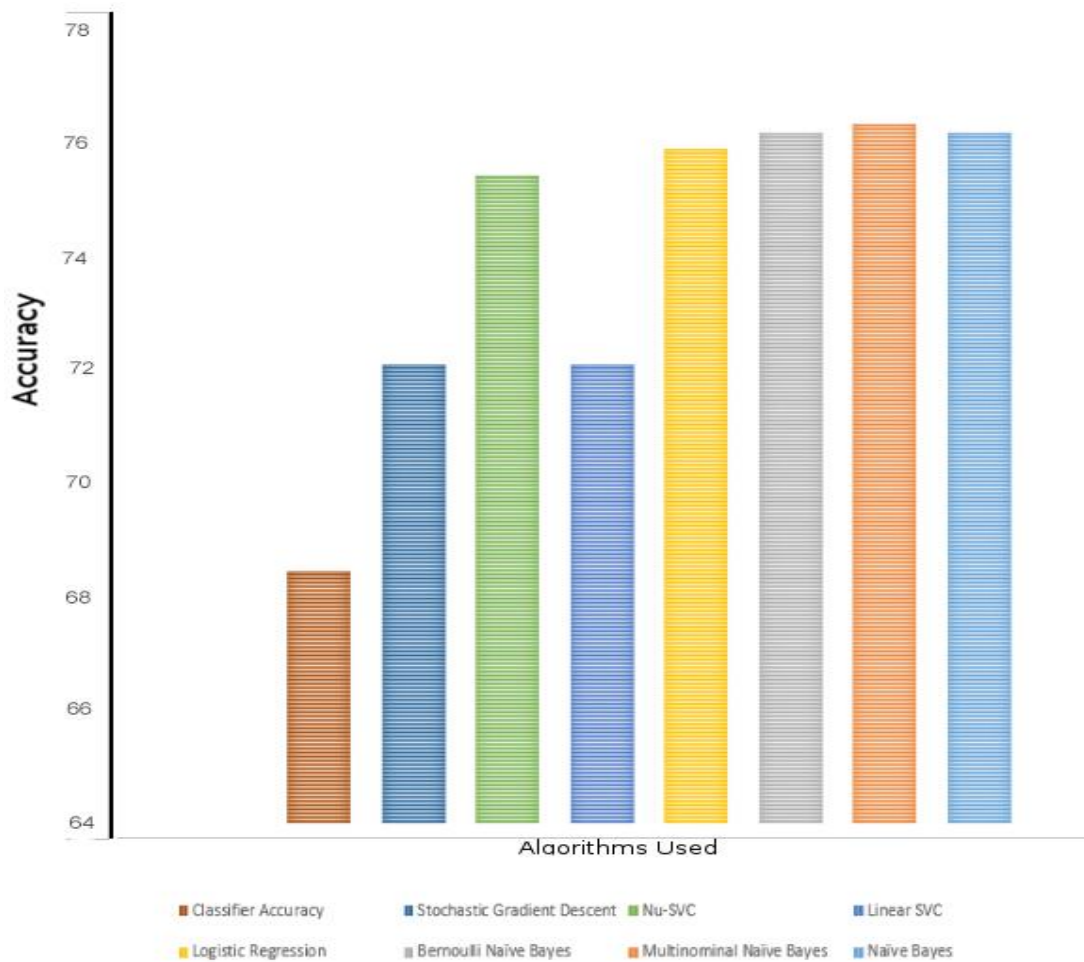
NuSVC\_classifier accuracy percent: 75.37764350453172

SGDClassifier accuracy percent: 72.05438066465257

RFClassifier accuracy percent: 68.42900302114803

ABClassifier accuracy percent: 67.82477341389728

voted\_classifier accuracy percent: 75.67975830815709



**Figure 3.4.1 Accuracy of classifiers using most common words approach**

### TF-IDF approach:

MNB\_classifier accuracy percent: 73.8742964353

BernoulliNB\_classifier accuracy percent: 73.5928705441

LogisticRegression\_classifier accuracy percent: 73.4052532833

LinearSVC\_classifier accuracy percent: 72.373358349

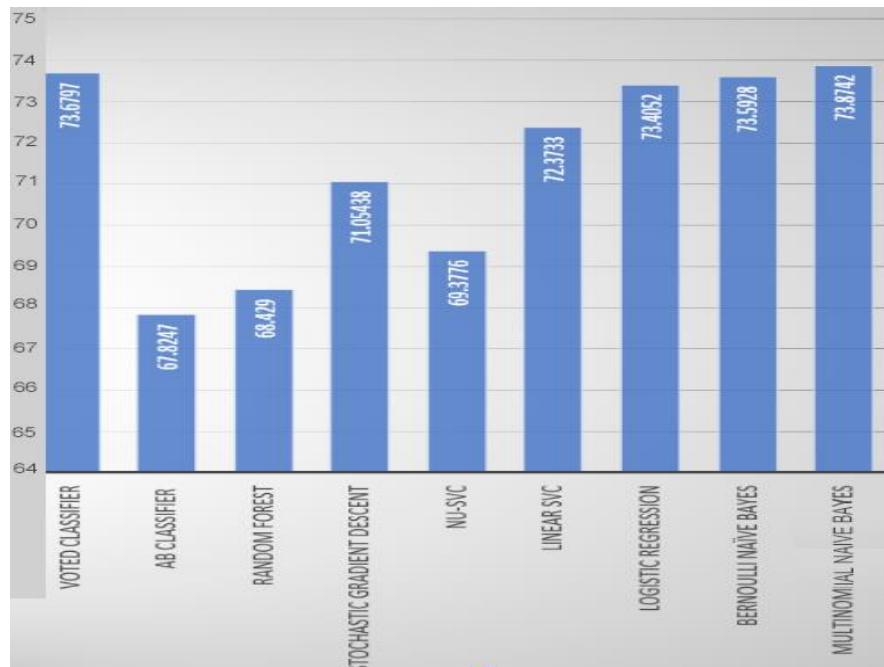
NuSVC\_classifier accuracy percent: 69.37764350453172

SGDClassifier accuracy percent: 71.05438066465257

RFClassifier accuracy percent: 68.42900302114803

ABClassifier accuracy percent: 67.82477341389728

voted\_classifier accuracy percent: 73.67975830815709



**Figure 3.4.1 Accuracy of classifiers using tf-idf approach**

### **3.5 Time metric**

#### **Most common words approach:**

Time taken to train NB classifier : 49.96313

Time taken to train MNB classifier : 52.196197

Time taken to train BernoulliNB classifier : 50.518579

Time taken to train Logistic Regression : 55.088875

Time taken to train LinearSVC\_classifier : 61.147708

Time taken to train NuSVC\_classifier : 287.193036

Time taken to train SGDCclassifier : 49.073194

Time taken to train RFclassifier : 48.13248

Time taken to train ABclassifier : 250.37365

Time taken to train Voting classifier : 251.744273

#### **TF-IDF approach:**

Time taken to train MNB classifier : 0.01385

Time taken to train BernoulliNB classifier : 0.005185

Time taken to train Logistic Regression : 0.306386

Time taken to train LinearSVC\_classifier : 0.025476

Time taken to train NuSVC\_classifier : 3.145843



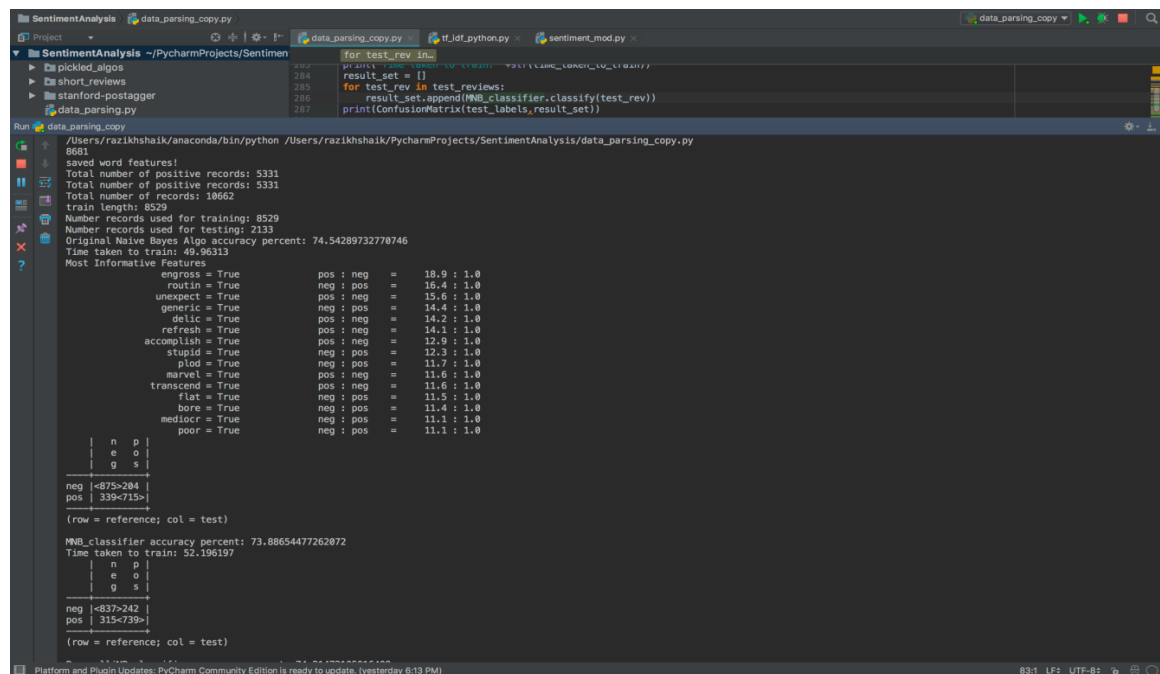
Time taken to train SGDCclassifier : 0.00558

Time taken to train RFclassifier : 0.547706

Time taken to train ABclassifier : 204.142836

Time taken to train Voting classifier : 223.646151

### 3.6 Screenshots



```
data_parsing_copy.py
/Users/razikhshaik/anaconda/bin/python /Users/razikhshaik/PycharmProjects/SentimentAnalysis/data_parsing_copy.py
8681
saved word features!
Total number of positive records: 3331
Total number of positive records: 3331
Total number of records: 18662
train length: 8529
Number records used for training: 8529
Number records used for testing: 2133
Original Naive Bayes Algo accuracy percent: 74.54289732770746
Time taken to train: 49.96313
Most Informative Features
engross = True          pos : neg = 18.9 : 1.0
routin = True           neg : pos = 16.4 : 1.0
unexpected = True       pos : neg = 15.6 : 1.0
generic = True          neg : pos = 14.4 : 1.0
delic = True            pos : neg = 14.2 : 1.0
refresh = True          pos : neg = 14.1 : 1.0
accomplish = True       pos : neg = 12.9 : 1.0
stupid = True           neg : pos = 12.3 : 1.0
plod = True             neg : pos = 11.7 : 1.0
marvel = True           pos : neg = 11.6 : 1.0
transcend = True        pos : neg = 11.6 : 1.0
flat = True             neg : pos = 11.5 : 1.0
bare = True            neg : pos = 11.4 : 1.0
mediocr = True          neg : pos = 11.1 : 1.0
poor = True            neg : pos = 11.1 : 1.0

| n p |
| e o |
| g s |
neg <875>204 |
pos | 339<715> |
(row = reference; col = test)
NB_classifier accuracy percent: 73.88654477262072
Time taken to train: 52.196157

| n p |
| e o |
| g s |
neg <837>242 |
pos | 315<739> |
(row = reference; col = test)
```

**Figure 3.6.1 Screenshot displaying prominent features**

```

data_parsing_copy.py
Project: SentimentAnalysis
data_parsing_copy.py
tfidf_python.py
sentiment_mod.py
data_parsing_copy.py

for test_rev in train_data_loader.get_test_data_loader():
    result_set = []
    for test_rev in test_reviews:
        result_set.append(RFC_classifier.classify(test_rev))
    print(ConfusionMatrix(test_labels,result_set))

Run data_parsing_copy

SGDClassifier accuracy percent: 72.9957885987173
Time taken to train: 49.073194
| n p |
| e o |
| g s |
neg <808>-271
pos <305>-749>
(row = reference; col = test)

RFCClassifier accuracy percent: 67.27613689639086
Time taken to train: 48.13248
| n p |
| e o |
| g s |
neg <852>-227
pos <471>-583>
(row = reference; col = test)

ABClassifier accuracy percent: 66.80731364275668
Time taken to train: 258.37365
| n p |
| e o |
| g s |
neg <768>-311
pos <397>-657>
(row = reference; col = test)

voted_classifier accuracy percent: 74.40225835161744
Time taken to train: 251.744273
| n p |
| e o |
| g s |
neg <856>-223
pos <323>-731>
(row = reference; col = test)

Classification: neg Confidence %: 77.77777777777779
Classification: pos Confidence %: 88.88888888888889
Classification: neg Confidence %: 88.88888888888889
Classification: pos Confidence %: 88.88888888888889

Platform and Plugin Updates: PyCharm Community Edition is ready to update. (yesterday 6:13 PM)
144:1 LF: UTF-8:

```

**Figure 3.6.2(a) Screenshot displaying accuracy of classifiers using most common words approach**

```

data_parsing_copy.py
Project: SentimentAnalysis
data_parsing_copy.py
tfidf_python.py
sentiment_mod.py
data_parsing_copy.py

for test_rev in train_data_loader.get_test_data_loader():
    result_set = []
    for test_rev in test_reviews:
        result_set.append(RFC_classifier.classify(test_rev))
    print(ConfusionMatrix(test_labels,result_set))

Run data_parsing_copy

(row = reference; col = test)

BernoulliNB_classifier accuracy percent: 74.21472105016489
Time taken to train: 50.518579
| n p |
| e o |
| g s |
neg <886>-195
pos <355>-692>
(row = reference; col = test)

LogisticRegression_classifier accuracy percent: 73.51148616971402
Time taken to train: 55.088875
| n p |
| e o |
| g s |
neg <825>-254
pos <311>-743>
(row = reference; col = test)

LinearSVC_classifier accuracy percent: 72.52695733708391
Time taken to train: 61.147708
| n p |
| e o |
| g s |
neg <811>-268
pos <318>-736>
(row = reference; col = test)

NuSVC_classifier accuracy percent: 72.94889826535396
Time taken to train: 287.193836
| n p |
| e o |
| g s |
neg <801>-278
pos <299>-755>
(row = reference; col = test)

SGDClassifier accuracy percent: 72.9957885987173

Platform and Plugin Updates: PyCharm Community Edition is ready to update. (yesterday 6:13 PM)
75:16 LF: UTF-8:

```

**Figure 3.6.2(b) Screenshot displaying accuracy of classifiers using most common words approach**

The screenshot shows the PyCharm IDE with a project named 'SentimentAnalysis'. The file explorer on the left shows a directory structure with 'pickled\_algos', 'short\_reviews', 'stanford-postagger', and 'data\_parsing.py'. The 'Run' console at the bottom displays the output of the 'data\_parsing\_copy.py' script. The script iterates through different word count ranges (e.g., neg [-808>271], pos [305<749]) and evaluates three classifiers: RFClassifier, ABCClassifier, and voted\_classifier. For each classifier, it reports the accuracy percent and the time taken to train. The output also includes confusion matrices for each classifier. The process finished with exit code 0.

```

Run data_parsing_copy
neg [-808>271 |
pos [305<749>]
(row = reference; col = test)
RFClassifier accuracy percent: 67.27613689639006
Time taken to train: 48.13248
| n p |
| e o |
| g s |
neg [-852>227 |
pos [471<583>]
(row = reference; col = test)
ABClassifier accuracy percent: 66.88731364275668
Time taken to train: 258.37365
| n p |
| e o |
| g s |
neg [-768>311 |
pos [397<657>]
(row = reference; col = test)
voted_classifier accuracy percent: 74.48225835161744
Time taken to train: 251.744273
| n p |
| e o |
| g s |
neg [-856>223 |
pos [323<731>]
(row = reference; col = test)
Classification: neg Confidence %: 77.77777777777779
Classification: pos Confidence %: 88.88888888888889
Classification: neg Confidence %: 88.88888888888889
Classification: neg Confidence %: 100.0
Classification: pos Confidence %: 88.88888888888889
Classification: neg Confidence %: 100.0
Process finished with exit code 0

```

**Figure 3.6.3 Screenshot displaying confidence of classifiers**

The screenshot shows the PyCharm IDE with a project named 'SentimentAnalysis'. The file explorer on the left shows a directory structure with 'pickled\_algos', 'short\_reviews', and 'tf\_idf.py'. The 'Run' console at the bottom displays the output of the 'tf\_idf.py' script. The script imports 'expanduser' from 'os.path' and 'StanfordPOSTagger' from 'nltk.tag.stanford'. It then evaluates several classifiers (MNBClassifier, BernoulliNBClassifier, LogisticRegressionClassifier, LinearSVCClassifier, NuSVCClassifier, SGDClassifier, RFClassifier, ABCClassifier, and VotingClassifier) using tf-idf. For each classifier, it reports the accuracy percent and the time taken to train. The output also includes confusion matrices for each classifier. The process finished with exit code 0.

```

Run tf_idf.py
/Users/razikhshaik/anaconda/bin/python /Users/razikhshaik/PycharmProjects/SentimentAnalysis/tf_idf.py
10
10
MNBClassifier accuracy percent: 73.8742964353
Time taken to train: 0.813885
[[783 283]
 [274 792]]
BernoulliNBClassifier accuracy percent: 73.5928705441
Time taken to train: 0.805185
[[787 279]
 [284 782]]
LogisticRegressionClassifier accuracy percent: 73.4852532833
Time taken to train: 0.386386
[[776 290]
 [277 789]]
LinearSVCClassifier accuracy percent: 72.373358349
Time taken to train: 0.625476
[[766 300]
 [289 777]]
NuSVCClassifier accuracy percent: 69.2387692388
Time taken to train: 3.145843
[[680 386]
 [270 796]]
SGDClassifier accuracy percent: 71.8574108818
Time taken to train: 0.80558
[[765 301]
 [299 767]]
RFClassifier accuracy percent: 68.574108818
Time taken to train: 0.547786
[[781 285]
 [385 681]]
ABClassifier accuracy percent: 67.4015089381
Time taken to train: 206.142836
[[724 342]
 [353 713]]
VotingClassifier accuracy percent: 73.6866791745
Time taken to train: 223.646151
[[798 260]
 [293 773]]
Process finished with exit code 0

```

**Figure 3.6.4 Screenshot displaying accuracy of classifiers using tf-idf**

## **4. Conclusion**

There has been data explosion in recent years and it has been reported that most of the data is unstructured data. It is necessary to process and understand the unstructured data for various purposes. The Web has changed from “read-only” to “read-write”, thus, opening doors to the collection of enormous data that portrays human opinions and sentiments. The need of sentiment analysis arises here to gain collective results on what the masses express. The classification of methods in sentiment analysis show that the technique majorly depends on what kind of data is being analyzed and thus every type has its own suitable technique. The future sentiment analysis and opinion mining techniques need wider common-sense approaches and are to be inspired by human thought process so that the results are more honest, accurate and parallel to the human psychology. We can assume that the future of sentiment analysis will plug the existing gaps in being able to interpret meaning.

## References

1. Uan Sholanbayev. *Sentiment Analysis on Movie Reviews*.  
<https://cseweb.ucsd.edu/~jmcauley/cse190/reports/sp15/041.pdf>
2. Liu, B. 2006. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, Springer
3. Nasukawa, T. & Yi, J. 2003. *Sentiment analysis: capturing favorability using natural language processing*. In Proceedings of the 2nd international conference on Knowledge capture, October 23–25, 2003. (pp. 70–77). Florida, USA.
4. <https://www.brandwatch.com/blog/understanding-sentimentanalysis/>
5. <https://www.lexalytics.com/technology/sentiment>
6. <http://web4.cs.ucl.ac.uk/staff/jun.wang/blog/2009/07/08/tf-idf/>
7. Chai, K.; H. T. Hn, H. L. Chieu; “*Bayesian Online Classifiers for Text Classification and Filtering*”, Proceedings of the 25th annual international ACM SIGIR conference on Research and Development in Information Retrieval, August 2002, pp 97-104
8. Jiawei Han, Micheline Kamber. 2003. *DATA MINING Concepts and*

## *Techniques*

9. [https://en.wikipedia.org/wiki/Bayes%27\\_theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem)
10. <http://blog.datumbox.com/machine-learning-tutorial-the-naivebayes-text-classifier/>
11. <https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayestext-classification-1.html>
10. <https://nlp.stanford.edu/IRbook/html/htmledition/thebernoulli-model-1.html>
12. Daniel Jurafsky & James H. Martin. 2016. *Speech and Language Processing*
13. Tobias Gunther & Lenz Furrer. *GU-MLT-LT: Sentiment Analysis of Short Messages using Linguistic Features and Stochastic Gradient Descent*  
<http://www.aclweb.org/anthology/S13-2054>
14. Bird, Steven, Edward Loper and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.
15. Harish, R., et al. "Lexical analysis-a brief study."
16. <http://www.cs.cornell.edu/people/pabo/movie-review-data/>