Sentiment Analysis on IMDb Movie Reviews using Navies-Bayes Classifier for the Problem of Classifying the Polarity of the Text

Prakash R[a] and Repana Reddy Sekhar [b*]
a: Department of Mathematics, RV college of Engineering, Bengaluru, India.

b*: Corresponding author: Repana Reddy Sekhar, RV college of Engineering, Bengaluru, India.

ABSTRACT:

Social networking sites frequently use brief text messages to convey a range of emotions. We can identify the different emotions that the crowd is expressing such as joy, sorrow, fear, worry, etc., by examining brief sentences. We can ascertain a reviewer's general sentiment or opinion on a movie by using IMDb sentiment analysis. The Problem presents a contemporary method for utilizing the Navies-Bayes classifier for sentiment analysis of movie reviews. The implementation of machine learning and prediction algorithms is made possible by the examination of a document as a set of numbers. Therefore, we could easily create avenues for machine learning algorithms to be used if there were a means to translate these words intended for human perception and interpretation into numbers.

Key words: Support vectors, Classification, Short texts, Microblog, Naïve-Bayes.

1. INTRODUCTION:

People's opinions have always had an impact on our daily lives, and this is especially true for those who like to broadcast everything on social media. Ideas and opinions of others have always affected our own opinions. There is now more activity in the areas of podcasting, blogging, tagging, contributing to RSS, social bookmarking, and social networking due to the proliferation of websites and applications [1]. The text plays a vital aspect in information shared, where users share their opinions on trending topics, politics, movie reviews, etc. in the form of Short Texts (ST). ST has certain challenges like identification of sarcasm, sentiment, use of slang words, etc. Therefore, it becomes important to understand short texts and derive meaningful insights from them, which is generally known as Sentiment Analysis (SA) [2]. Sentiment analysis or opinion mining is the computational treatment of opinions, sentiments and subjectivity of text.

Because sentiment analysis can examine many documents or data at once, something that would take more time to perform manually, political parties, banking and various business sectors employ it extensively. Businesses in the business sector use SA to develop new strategies based on input from their customers. SA was crucial to the 2016 US Presidential Elections. On microblogs like Facebook and Twitter, users discussed what they liked and didn't like about a certain political party. Candidates edited their tweets in response to the analysis of those blogs. Reviews are short texts that generally express an opinion about movies or products. These reviews play a vital role in the success of movies or sales of the products [3-5].

People generally investigate blogs, review sites like IMDb to know about movie cast, crew, review and ratings. SA on movie reviews makes the task of opinion summarization easier by extracting the sentiment expressed by the reviewer [6]. To learn about the cast, crew, reviews, and ratings of a movie, people typically check blogs and review websites like IMDb. By removing the reviewer's sentiment, SA on movie reviews simplifies the process of summarizing opinions. SA's primary responsibilities are preprocessing, feature extraction, selection, classification, and, finally, outcome assessment. Preprocessing is a process of preparing the raw data and making it suitable form, Feature extraction identifies the most discriminating characteristics from raw data. Classification is used to categorize data into distinct classes [7-14].

The process of Sentiment Analysis involves the construction of the input vector space from the existing document vector space. Mainly there are two approaches to carry out vector space mapping. Machine learning based or statistical based feature extraction methods are widely used because extraction of features is done by applying statistical measures directly. Earlier works on sentiment classification using machine learning approaches were carried out by many researchers clearly identifies and distinguishes between a positive review and a negative review. Sentiment analysis has been thoroughly studied utilizing a variety of machine learning methods that deal with classification models [15-20].

Review of Tweets is a growing research area in the field of Natural Language Processing. In the past decade, new forms of communication, such as microblogging and text messaging have emerged and become ubiquitous. While there is no limit to the range of information conveyed by tweets and texts, often these short messages are used to share opinions and sentiments that people have about what is going on in the world around them. Tweets and texts are short: a sentence or a headline rather than a document. The language used is very informal, with creative spelling and punctuation, misspellings, slang, new words, URLs, and genre-specific terminology and abbreviations, such as, RT for "re-tweet" and # hashtags, which are a type of tagging for Twitter messages. Another aspect of social media data such as Twitter messages is that it includes rich structured information about the individuals involved in the communication. For example, Twitter maintains information of who follows whom and re-tweets and tags inside of tweets provide discourse information [21-23].

Beyond this, software has been designed in such way that it implements our algorithms in faculty feedback form. It is a common practice for people to simply enter any rating from 0 to 5 just to get rid of the form filling hassle. If everyone is required to enter text feedback rather, it will reflect more genuine sentiments. The problem with text feedback is that the teachers cannot keep reading hundreds of texts feedback. Our feedback form computes a rating of each text and returns and overall average of all ratings. Thus, we increase the genuinity of feedback forms while keeping the workload of teachers low.

In this paper, deep learning is being used for sentimental analysis by transforming it to a binary classification problem so that convolution neural network (CNN) can be applied for classification and segmentation. In this report, we look at the various challenges and applications of sentiment analysis and discussed in details various approaches to perform a computational treatment of sentiments and opinions by making use of various supervised or data-driven techniques. Various supervised or data-driven techniques to SA like Naive Byes, Maximum Entropy, support vector machine (SVM), and voted Perceptron's are already available in literature [24-30]. Various researchers testing new features and classification techniques often just compare their results to base-line performance. There is a need for proper and formal comparisons between these results arrived through different features and classification techniques to select the best features and most efficient classification techniques for applications. In view of this, the present report mainly focuses on Navies-Bayes Classifier by considering the problem of classifying the polarity of the given text at the document or sentence or feature/ aspect level (positive, negative or neutral) by applying machine learning algorithm. The objectives of this work are as follows:

1. Preliminary Sentiment Analysis on the movie reviews
2. Applying the unigram, bigram model
3. Machine Learning Algorithm.

1.1. Proposed Architecture

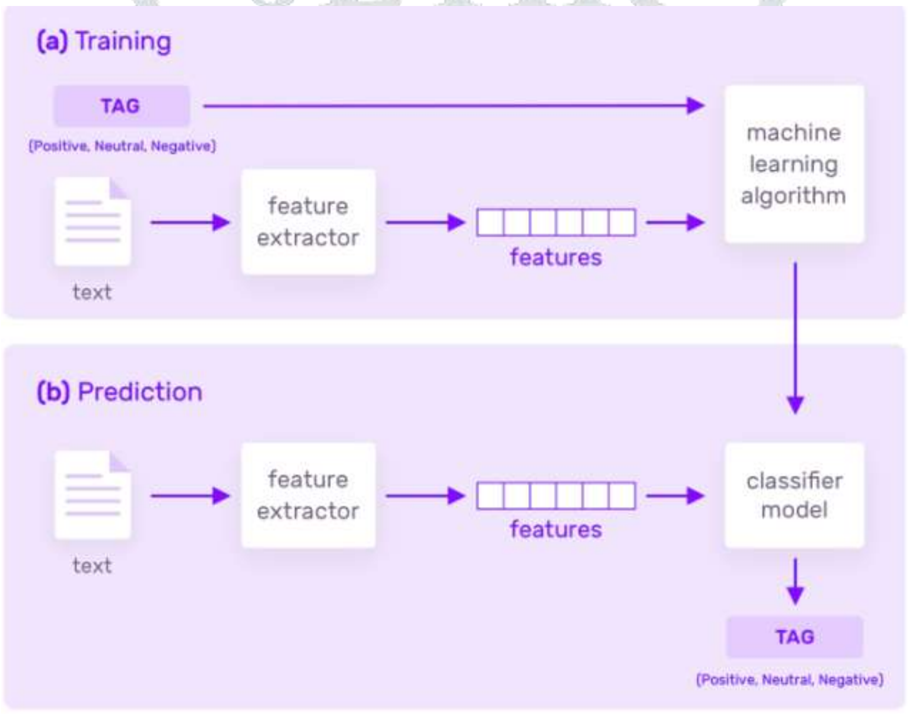


Fig. 1: Flow chart of the sentiment analysis

The Training and Prediction Processes

In the training process (a), model learns to associate a particular input (i.e. a text) to the corresponding output (tag) based on the test samples used for training. The feature extractor transfers the text input into a feature vector. Pairs of feature vectors and tags (e.g. positive, negative, or neutral) are fed into the machine learning algorithm to generate a model.

In the prediction process (b), the feature extractor is used to transform unseen text inputs into feature vectors. These feature vectors are then fed into the model, which generates predicted tags (again, positive, negative, or neutral).

Feature Extraction from Text

The first step in a machine learning text classifier is to transform the text into a numerical representation (vector). Usually, each component of the vector represents the frequency of a word or expression in a predefined dictionary (e.g. a lexicon of polarized words). This process is known as feature extraction or text vectorization and the classical approach has been bag-of-words or bag-of-n grams with their frequency.

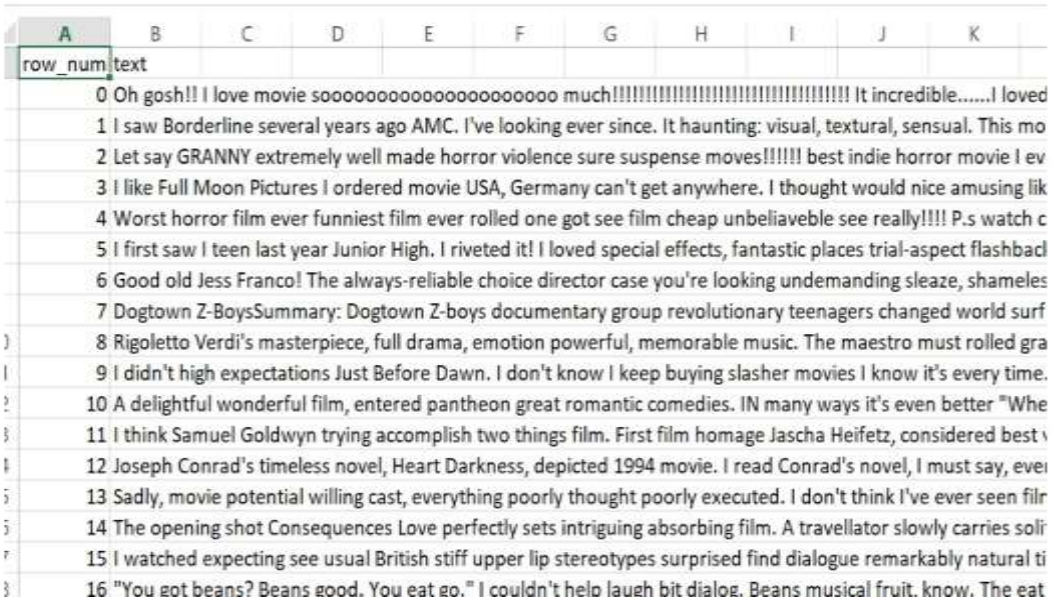
More recently, new feature extraction techniques have been applied based on word embeddings (also known as *word vectors*). This kind of representations makes it possible for words with similar meaning to have a similar representation, which can improve the performance of classifiers.

2. DATA PREPROCESSING:

Before performing any algorithm, we have to make sure to clean up the data, in order to make it easier to process. Also, by finding and removing noise words in advance, we can increase greatly the accuracy of our algorithms.

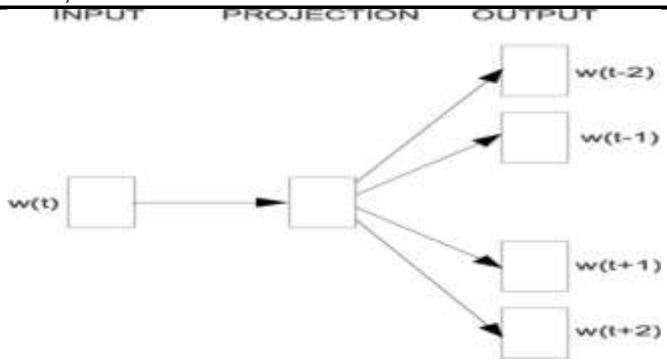
2.1 Cleaning the data

The IMDb reviews contains html tags which do not serve any purpose for detecting sentiment, It is decided to remove punctuation whatsoever, even if this means that we get rid of emoticons (there are very few of them anyway), but it makes it easier for us to handle. Porter stemming algorithm, which helps to replace every word with its root, and so words like cats and cat, or running and run, become the same. This has been shown to improve classification accuracy in sentiment analysis tasks.



A	B
row_num	text
0	Oh gosh!! I love movie soooooooooooooooooooo much!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! It incredible......I loved
1	I saw Borderline several years ago AMC. I've looking ever since. It haunting: visual, textural, sensual. This mo
2	Let say GRANNY extremely well made horror violence sure suspense moves!!!!!! best indie horror movie I ev
3	I like Full Moon Pictures I ordered movie USA, Germany can't get anywhere. I thought would nice amusing lik
4	Worst horror film ever funniest film ever rolled one got see film cheap unbeliaveble see really!!!! P.s watch c
5	I first saw I teen last year Junior High. I riveted it! I loved special effects, fantastic places trial-aspect flashbacl
6	Good old Jess Franco! The always-reliable choice director case you're looking undemanding sleaze, shameles
7	Dogtown Z-BoysSummary: Dogtown Z-boys documentary group revolutionary teenagers changed world surf
8	Rigoletto Verdi's masterpiece, full drama, emotion powerful, memorable music. The maestro must rolled gra
9	I didn't high expectations Just Before Dawn. I don't know I keep buying slasher movies I know it's every time.
10	A delightful wonderful film, entered pantheon great romantic comedies. IN many ways it's even better "Whe
11	I think Samuel Goldwyn trying accomplish two things film. First film homage Jascha Heifetz, considered best
12	Joseph Conrad's timeless novel, Heart Darkness, depicted 1994 movie. I read Conrad's novel, I must say, ever
13	Sadly, movie potential willing cast, everything poorly thought poorly executed. I don't think I've ever seen filr
14	The opening shot Consequences Love perfectly sets intriguing absorbing film. A travellator slowly carries soli
15	I watched expecting see usual British stiff upper lip stereotypes surprised find dialogue remarkably natural ti
16	"You got beans? Beans good. You eat go." I couldn't help laugh bit dialog. Beans musical fruit. know. The eat

Figure 2.1a Data-Set



2.2 Stop word Removal

Stop words are words which carry a connecting function in the sentence, such as prepositions, articles, etc. There is no definite list of stop words, but some search machines, are using some of the most common, short function words, such as the, is, at, which, and on. They can be removed since they have a high frequency of occurrence in the text, but do not affect the final sentiment of the sentence. For the purpose of stop-words are stored in a file and using these words, the stop words are removed from the dataset.

2.3 Unigram, bigram model

A statistical language model is a probability distribution over sequences of words. Given such a sequence, say of length m, it assigns a probability to the whole sequence $P\ (w_1, \dots, w_m)$.

The language model provides context to distinguish between words and phrases that sound similar. For example, in American English, the phrases "recognize speech" and "wreck a nice beach" sound similar, but mean different things.

Data sparsity is a major problem in building language models. Most possible word sequences are not observed in training. One solution is to make the assumption that the probability of a word only depends on the previous n words. This is known as an n-gram model or unigram model when n = 1. The unigram model is also known as the bag of words model.

Estimating the relative likelihood of different phrases is useful in many natural language processing applications, especially those that generate text as an output. Language modeling is used in speech recognition, machine translation, part-of-speech tagging, parsing, Optical Character Recognition, handwriting recognition, information retrieval and other applications.

In speech recognition, sounds are matched with word sequences. Ambiguities are easier to resolve when evidence from the language model is integrated with a pronunciation model and an acoustic model.

Language models are used in information retrieval in the query likelihood model. There a separate language model is associated with each document in a collection. Documents are ranked based on the probability of the query Q in the document's language model $P\ (Q \mid M_d)$. Commonly, the unigram language model is used for this purpose.

A unigram model can be treated as the combination of several one-state finite automata. It splits the probabilities of different terms in a context in the following from.

$$P_{uni}\ (t_1 t_2 t_3\ ) \ = P\ (t_1)\ P\ (t_2)\ P\ (t_3)$$

The probability generated for a specific query is calculated as

$$P\ (\text{query}) = \prod_{term\ in\ query} P\ (term)$$

The terms **bigram** and **trigram** language models denote *n*-gram models with $n = 2$ and $n =$

3, respectively.

2.1 TF_IDF

Tf-Idf stands for term frequency, inverse document frequency. And it is a very useful technique used mainly in information retrieval in order to rank how important a keyword is to a given document in a corpus1 . Returning to the word movie. This made us think that we still had garbage, and we had the idea that maybe a word can be very important to the whole set of reviews, but not to any single review by itself, and thus we needed a way of computing which words were important to which reviews. After searching a bit we came with the concept of tf-idf, which solves precisely our problem.

It works as follows, we want to compute the function w(w, d) for every word w document d pair, this will give how important is the word for indexing the document. We have the two auxiliary functions tf(w, d), which counts how important is the word for the document.

tf(w, d) = 1 + log(Number of occurrences of w in d)

This intuitively gives how important is that word in the document, but it doesn't take into account the possibility of a word belonging to every document (which will render it useless for indexing), and thus we need a balance term idf(w, D), which is defined over the whole corpus D, and is given by

idf(w, D) = log |D| |{d ∈ D : w ∈ D}|

The more documents w is in, the lower it's idf score, which is what we want. So now, we get that f(w, D) = tf(w, d) ∗ idf(w, D)

And thus a word is important if it is frequent in the document, but if it is not to frequent among all other documents.

3. METHODOLOGY:

The methodology to achieve the above-mentioned objectives is stated below:

1. To download data set of reviews from imdb handles :

2. To preprocess the data:

   The downloaded sets of reviews were then put into a CSV file using pandas library. unigram, bigram modelling was used to convert the dataset into useful data points. It used correlation mechanism to realize the importance of a sentiment in the sentence. Then if a single word is found that does not fit in the ongoing sentiment, it is ignored based on the concept of concordant readings.

3. To make the machine learning model :

   The formed vectors were then used to train a model. A stochastic gradient descent algorithm was found to best suit the requirement of minimum accuracy.

4. To verify the accuracy of the prediction:

   The accuracy was measured by use of pandas library to compare the predicted sentiment with manual ratings. Use of various algorithms suggested that best is the use of SGD for NLP tasks.

Tools used: Anaconda.

Dataset size: 25000 reviews

Format of dataset: Uncleaned semi-structured

4. IMPLEMENTATION DETAILS OF THE WORK:

#Preprocessing the dataset

```
def imdb_data_preprocess(inpath, outpath="./", name="imdb_tr.csv", mix=False): import pandas as pd
    from pandas import DataFrame, read_csv import os
    import csv
    import numpy as np
    stopwords = open("stopwords.en.txt", 'r' , encoding="ISO-8859-1").read() stopwords =
    stopwords.split("\n")
    indices = [] text = [] rating = [] i =  0
    for filename in os.listdir(inpath+"pos"):
        data = open(inpath+"pos/"+filename, 'r' , encoding="ISO-8859-1").read() data =
        remove_stopwords(data, stopwords)
        indices.append(i) text.append(data) rating.append("1") i = i + 1
    for filename in os.listdir(inpath+"neg"):
        data = open(inpath+"neg/"+filename, 'r' , encoding="ISO-8859-1").read() data =
        remove_stopwords(data, stopwords)
        indices.append(i) text.append(data) rating.append("0")
        i = i + 1
    Dataset = list(zip(indices,text,rating))
    if mix:
        np.random.shuffle(Dataset)
    df = pd.DataFrame(data = Dataset, columns=['row_Number', 'text', 'polarity']) df.to_csv(outpath+name,
    index=False, header=True)
    pass
```

#Removing the stopwords

```
def remove_stopwords(sentence, stopwords): sentencewords = sentence.split()
        resultwords = [word for word in sentencewords if word.lower() not in stopwords] result = '
        '.join(resultwords)
        return result
```

#Unigram model

```
def unigram_process(data):
        from sklearn.feature_extraction.text import CountVectorizer vectorizer = CountVectorizer()
        vectorizer = vectorizer.fit(data) return vectorizer
```

#Bigram model

```
def bigram_process(data):
        from sklearn.feature_extraction.text import CountVectorizer vectorizer =
        CountVectorizer(ngram_range=(1,2)) vectorizer = vectorizer.fit(data)
        return vectorizer
```

#Retrieve the data from the dataset

```
def retrieve_data(name="imdb_tr.csv", train=True): import pandas as pd
        data = pd.read_csv(name,header=0, encoding = 'ISO-8859-1') X = data['text']
        if train:
                Y = data['polarity'] return X, Y
        return X
```

#Finding the accuracy

```
def accuracy(Ytrain, Ytest):
        assert (len(Ytrain)==len(Ytest))
        num = sum([1 for i, word in enumerate(Ytrain) if Ytest[i]==word]) n = len(Ytrain)
        return (num*100)/n
```

SOFTWARE TESTING AND VALIDATION

The first part of our approach is to consider the problem domain and try to determine equivalence classes based on the properties of real-world data sets. Testing was done at every stage. We particularly look for traits that may not have been considered by the algorithm designers, such as data set size, the potential ranges of attribute and label values, and what sort of precision is expected when dealing with floating point numbers. The data sets of interest are very large, both in terms of the number of attributes (hundreds) and the number of examples (tens of thousands). The label could be any non-negative integer, although it was typically a 0 (indicating that there was no device failure) or 1 (indicating that there was), and rarely was higher than 5 (indicating five failures over a given period). The attribute values were either numerical or categorical. Many non-categorical attributes had repeated values and many values were missing, raising the issues of breaking “ties” during sorting and handling unknowns. We do not discuss categorical attributes further (because we found no relevant bugs).

The second element to our approach to creating test cases was to look at the algorithm as it is defined (in pseudocode, for instance) and inspect it carefully for imprecisions, particularly given what we knew about the real-world data sets as well as plausible "synthetic" data sets. This would allow us to speculate on areas in which flaws might be found, so that we could create test sets to try to reveal those flaws.

The last part of our approach to generating test cases for ML algorithms is to look at their runtime options and see if those give any indication of how the implementation may manipulate the input data and try to design data sets and tests that might reveal flaws or inconsistencies in that manipulation.

To facilitate our testing, we created a set of utilities targeted at the ML algorithms we investigated. The utilities currently include: a data set generator; tools to compare a pair of output models and rankings; several trace options inserted into the ML implementations; and tools to help analyze the intermediate results indicated by the traces. Using our testing approach, we devised the following basic equivalence classes: small vs. large data sets; repeating vs. non-repeating attribute values; missing vs. non-missing attribute values; repeating vs. non-repeating labels; negative labels vs. non-negative only labels; predictable vs. non-predictable data sets and combinations thereof. These equivalence classes were then used to parameterize the test case selection criteria applied by our data generator tool to automate creation of appropriate input data sets.

5. RESULTS AND ANALYSIS:

The implementations are combined into batches of size 32 for each epoch.

Accuracy: Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions the model got right which is defined as follows.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

Where, TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives. However, accuracy is not enough when dealing with class imbalanced datasets, thus other metrics are considered as well.

AUC (ROC) Score: The Receiver Operating Characteristic (ROC) Curve is a very useful tool when predicting the probability of a binary outcome. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0. Put another way, it plots the false alarm rate versus the hit rate.

The True Positive Rate (TPR) is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. It describes how good the model is at predicting the positive class when the actual outcome is positive. Eq. 6.2 defines the true positive rate.

$$TPR = \frac{TP}{TP + FN} \qquad (6.2)$$

The False Positive Rate (FPR) is calculated as the number of false positives divided by the sum of the number of false positives and the number of true negatives. It is also called the false alarm rate as it summarizes how often a positive class is predicted when the actual outcome is negative. Eq. 6.3 defines the false positive rate.

$$FPR = \frac{FP}{FP + TN} \qquad (6.3)$$

The Area Under the Curve (AUC) score can be used as a summary of the model skill. Generally, skillful models are represented by curves that bow up to the top left of the plot.

Precision & Recall: Precision is a ratio of the number of true positives divided by the sum of the true positives and false positives. It describes how good a model is at predicting the positive class. Eq. 6.4 defines the precision metric.

$$Precision = \frac{TP}{TP + FP} \qquad (6.4)$$

positive calculated as the ratio of the number of true positives divided by the sum of the true positives and the false negatives. Eq. 6.5 defines the recall metric.

$$Recall = Sensitivity = \frac{TP}{TP + FN} \qquad (6.5)$$

A precision-recall curve is a plot of the precision (y-axis) and the recall (x-axis) for different thresholds, much like the ROC curve. A skillful model is represented by a curve that bows towards [1.0,1.0] above the flat line of no skill.

F1 score: that calculates the harmonic mean of the precision and recall (harmonic mean because the precision and recall are ratios).

Average precision: It summarizes the weighted increase in precision with each change in recall for the thresholds in the precision-recall curve.

Classifier	**Training Accuracy (%)**	**Validation Accuracy (%)**
Random Forest	72.69	55.22
Support Vector Machine	73.76	76
Stochastic Gradient Process	79	87
Naïve Baye's	**65.3**	**84**

Table 5.1: Accuracy Scores for the different classifiers

From the observations, it can be inferred that **Stochastic Gradient Descent** performs the best as the final classifier of the Sentiment Classification module.

Classifier	AUC(ROC)Score	F1 - Score	Average Precision
Random Forest	0.879	0.944	0.927
Support Vector Machine	0.782	0.641	0.905
Naïve Baye's	0.86	0.76	0.87
Stotastic Gradient Process	0.853	0.937	0.936

Table 5.2: AUC (ROC), F1 and average precision scores for the different classifiers

AUC score, F1 score, and average precision scores are computed and represented in Table 5.2. Random forest outperforms the others in terms of all the measures, performing slightly better than SVM and Decision trees. SVM and Decision Trees show comparable performance in terms of metrics taken in Table 5.2, however SVM performed starkly better in terms of accuracy. From the observations, it can be inferred that Stochastic Gradient Descent performs the best as the final classifier of the Sentiment Classification module in the project.

6. CONCLUSION:

The task of sentiment analysis, especially in the domain of micro-blogging, is still in the developing stage and far from complete. Due to the limited number of characters, huge dimensional features and sparseness, which increases complication. Right now, only the very simplest unigram model is considered, and these models are improved by adding extra information like closeness of the word with a negation word. We could specify a window prior to the word (a window could for example be of 2 or 3 words) under consideration and the effect of negation may be incorporated into the model if it lies within that window. The closer the negation word is to the unigram word whose prior polarity is to be calculated, the more it should affect the polarity.

Various models are compared, found that combination of unigram and Naïve Bayes model more effective in terms of ease of training and understanding. To demonstrate the effectiveness of the proposed work, we used four different classifiers such as Random Forest, SVM, SGM and Naïve Bayes on IMDb movie review dataset. In subsequent research, we want to improve the classification accuracy by adding additional lexical characteristics to the feature subset.

Apart from this, we are currently only focusing on unigrams and the effect of bigrams and trigrams may be explored by incorporating parts of speech within the unigram models. One more feature that is worth exploring is whether the information about relative position of word in a tweet has any effect on the performance of the classifier. So, we can attempt to perform separate sentiment analysis on tweets that only belong to class instead of focusing on general sentiment analysis. We can attempt to model human confidence in our system. So that the effects of human confidence can be visualized in sentiment analysis.

ACKNOWLEDGMENT: The authors express their gratitude towards the institution.

REFERENCES:

[1] A. J. Irías Tejeda, R. Castro Castro, "Algorithm of Sentiment Detection by Computational Vision", in IEEE International Conference on Electronics, Communications and Computers (CONIELECOMP), 124-128, 2019.
[2] I. Chetviorkin, P. Braslavskiy, N. Loukachevich, "Sentiment Analysis Track at ROMIP 2011," In Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialog 2012", Bekasovo, 1-14, 2012.
[3] Tirath Prasad Sahu, Sanjeev Ahuja, Sentiment Analysis of Movie Reviews: A study on Feature Selection & Classification Algorithms https://ieeexplore.ieee.org/document/7522583 :2016
[4] T. Mikolov, K. Chen, G. Corrado, J. Dean, "Efficient Estimation of Word Representations in Vector Space," In Proc. of Workshop at ICLR, 2013.
[5] P. Bo and L. Lee, "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts," In Proceedings of the ACL, 2004
[6] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," In European Conference on Machine Learning (ECML), Springer Berlin/Heidelberg, 137-142, 1998.
[7] P.D. Turney, "Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews," Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02), Philadelphia, Pennsylvania, 417-424, 2002.
[8] A. Go, R. Bhayani, L. Huang, "Twitter Sentiment Classification Using Distant Supervision," Technical report, Stanford. 2009.
[9] J. Furnkranz, T. Mitchell, and E. Riloff, "A Case Study in Using Linguistic Phrases for Text Categorization on the WWW," In AAAI/ICML Workshop on Learning for Text Categorization, 5-12, 1998.
[10] M.F. Caropreso, S. Matwin, F. Sebastiani, "A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization," Text databases and document management: Theory and practice, 78-102, 2001.
[11] C. D. Santos and M. Gatti. "Deep convolutional neural networks for sentiment analysis of short texts." In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, 69-78, 2014.
[12] A. Ortigosa, J. M. Martín, and R. M. Carro. "Sentiment analysis in Facebook and its application to e-learning." Computers in Human Behavior, 527-541, 2014.
[13] R. Ahmad, A. Pervaiz, P. Mannan, and F. Zaffar. "Aspect Based Sentiment Analysis for Large Documents with Applications to US Presidential Elections 2016." Social Technical and Social Inclusion Issues (SIGSI), 2017.
[14] K. Xu, S. S. Liao, J. Li, and Y. Song. "Mining comparative opinions from customer reviews for Competitive Intelligence." Decision support systems, 743-754, 2011.
[15] A. Tripathy, A. Agrawal, and S.K. Rath. "Classification of sentiment reviews using n-gram machine learning approach." Expert Systems with Applications, 117-126, 2016.
[16] M. E. Moussa, E. H. Mohamed, and M. H. Haggag. "A survey on Opinion Summarization Techniques for Social Media." Future Computing and Informatics Journal, 2018.
[17] I. Hemalatha, G. P. S. Varma, and A. Govardhan. "Preprocessing the informal text for efficient sentiment analysis." International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) 1, 58-61, 2012.
[18] A. G. Prasad, S. Sanjana, S. M. Bhat, and B. S. Harish. "Sentiment analysis for sarcasm detection on streaming short text data." In Knowledge Engineering and Applications (ICKEA), 2017, 2nd International

Conference, IEEE, 1-5, 2017.

[19] J. Brooke, M. Tofiloski, and M. Taboada. “Cross-linguistic sentiment analysis: From English to Spanish.” In Proceedings of the international conference RANLP-2009, 50-54, 2009.

[20] P. Melville, W. Gryc, and R. D. Lawrence. “Sentiment analysis of blogs by combining lexical knowledge with text classification.” In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 1275-1284, 2009.

[21] O. Kolchyna, T. T. P. Souza, P. Treleaven, and T. Aste. “Twitter sentiment analysis: Lexicon method, machine learning method and their combination.” , 2015.

[22] Y. Bao, C. Quan, L. Wang, and F. Ren. “The role of pre-processing in twitter sentiment analysis.” In International Conference on Intelligent Computing, Springer, 615-624, 2014.

[23] G. Gautam, and D. Yadav. “Sentiment analysis of twitter data using machine learning approaches and semantic analysis.” In Contemporary computing (IC3), 2014 seventh international conference, IEEE, 437-442, 2014.

[24] A. S. Manek, P. D. Shenoy, M. C. Mohan, and K. R. Venugopal. “Aspect term extraction for sentiment analysis in large movie reviews using Gini Index feature selection method and SVM classifier.” World Wide Web, 135-154, 2017.

[25] A. Kennedy and D. Inkpen. “Sentiment classification of movie reviews using contextual valence shifters.” Computational intelligence, 110-125, 2006.

[26] M. Z. Asghar, A. Khan, S. Ahmad, and F. M. Kundi. “A review of feature extraction in sentiment analysis.” Journal of Basic and Applied Scientific Research, 181-186, 2012.

[27] A. Sharma and S. Dey. “A comparative study of feature selection and machine learning techniques for sentiment analysis.” In Proceedings of the 2012 ACM research in applied computation symposium, ACM, 1-7, 2012.

[28] B. Pang, L. Lee, and S. Vaithyanathan. “Thumbs up?: sentiment classification using machine learning techniques.” In Proceedings of the ACL-02 conference on Empirical methods in natural language processing- Volume 10, Association for Computational Linguistics, 79-86, 2002.

[29] M. S. Mubarok, Adiwijaya, and M. D. Aldhi. “Aspect-based sentiment analysis to review products using Naïve Bayes.” In AIP Conference Proceedings, 1-8, 2017.

[30] H. M. Keerthi Kumar, B. S. Harish and H. K. Darshan, “Sentiment Analysis on IMDb Movie Reviews using Hybrid feature Extraction Method”, Intl J. Interactive Multimedia and Artificial Intelligence, 109-114, 2018.