



Classifying the Sentiment of IMDB Reviews Through Machine Learning Models

Pottem Samhith Kumar¹, Prateek Mudali², Dr. K.R. Usha Rani³, S. Praveen⁴

Student, Department of ECE, R.V. College of Engineering, Bangalore, India^{1,2}

Associate Professor, Department of ECE, R.V. College of Engineering, Bangalore, India³

Assistant Professor, Department of ECE, R.V. College of Engineering, Bangalore, India⁴

Abstract: Since the mid-90s, utilization of the web has augmented in different structures. Individuals are conveying utilizing different appearances. With this humungous development of web traffic, distinctive online web-based platforms, for example, Facebook, Twitter, LinkedIn, and so forth, are likewise getting popular. Billions of clients are imparting their insight on various perspectives on websites, for example, Facebook, Tumbler, Twitter, glint, LinkedIn, and so forth. Twitter is the most well-known small-scale blogging and person to person communication administration, which gives an office to clients to sharing, conveying, and translating 140 words' posts known as a tweet. Twitter has 320 Million month to month dynamic clients. It is open through site interface, SMS, or portable gadgets. Characteristic language handling is additionally assuming a major job and can be utilized by the suppositions communicated. In this project, the authors are trying to analyze the users' sentiment on tweets collected from different sources on the IMDB movie reviews. The project successfully predicts if the sentiment of the review will be of a negative or positive polarity. In this paper, the authors came across different methods of implementations and identified various techniques through which features of the reviews can be defined considering it either as single entities or a whole document. The authors of this project implemented a technique called distributed representation of sentences and documents, which ultimately gave them the highest accuracy among all the models they implemented.

Keywords: Machine Learning, Sentiment Analysis, IMDB, Random Forest, ANN, SVM.

I. INTRODUCTION

Sentiment Analysis is a contextual text mining technique which finds and explores information in the source material to help the business by understanding the sentiment of the product socially, brand/service while surveilling the online conversation. However, the social media stream analysis is restricted to just fundamental sentiment analysis and metric based on counts. This is like scratching the surface and neglecting the most valuable insights about the data that had been waiting to be found in the analysis [1]. It is also referred to as opinion mining and has various uses in business Intelligence applications, including questions regarding why a certain product is sold more. What does the customer think about the product, etc.?

It also has several cross-domain applications in various fields, including politics, lawmaking, sociology, psychology, etc. and the most popular text classification tool which tests incoming messages and tells whether the sentiment is neutral, positive, or negative. In this paper, the authors include a sentence of their choice and estimate the underlying sentiment insights that are waiting to be found.

Sentiment Analysis comes with several challenges as people express an opinion in complex ways. Lexical content can be misleading in some of the text. Intra-textual reversals, negations, topic changes are common [2]. Rhetorical modes such as irony, implication, sarcasm, etc. will make it a more tedious task.

II. LITERATURE REVIEW

Sentiment Analysis of Twitter data has been widely researched in the past few years. Efthymios Kouloumpis, Theresa Wilson, and a few other researchers together found tri-polar classification of tweets, and they concluded that Parts of speech tagging doesn't contribute much in this aspect. Tajinder Singh and Madhu Kumari from the National Institute of Technology, Hamirpur, proposed a well-defined flow for the text preprocessing and focused on finding the impact of slang words in sentiment analysis [3].

Andrew L. Maas, Raymond E. Daly, and other researchers from Stanford University used a mix of supervised and unsupervised techniques to learn word vectors by capturing rich sentiment content. A broad data collection of film reviews was also added to act as a benchmark in this field. Quoc Le and Tomas Mikolov from Google Inc proposed



Paragraph Vector, an unsupervised algorithm that learns fixed-length features from paragraphs, sentences, and documents. In this paper, the authors have studied these research papers in- depth [4].

Richard Socher, Jeffrey Pennington, Eric H.Huang and rest of the authors presented a new Artificial Intelligence framework which depends upon recursive autoencoders for sentence-level assumption of sentiment label distributions. Vector space representations for multi-word phrases has been learnt by them. The authors in [5] also assess the model's capacity to foresee sentiment distributions on a novel dataset dependent on disclosure that from the project experience. The dataset comprises of individual stories of the user that is annotated with various labels and then on summing up, forms a multinomial distribution which accurately catches emotional responses.

The paper [6] depicts the framework presented for the Sentiment Analysis of SEMEVAL 2014 (Twitter task) and explicitly the Message Polarity Classification subtask. Rafael Michael Karampetsos, John Pavlopoulos and prodromos Malakasis utilized a 2-stage pipeline approach adopting a linear SVM classifier at every stage.

A. METHODOLOGY ADOPTED

The approach followed in this paper is as follows:

i. Data Collection:

Complete information is gathered from the Andrew Maas, Stanford University. The dataset is for twofold assessment grouping containing significantly a greater number of information than the other benchmark datasets. He gives a lot of 25K profoundly polar movie surveys to train the model and 25K audit information to test the model [5]. There is extra unlabeled information for use, too, which can be used for other binary classifications. Crude content and a previously processed and handled bag of words groups are also collected.

ii. Data Preprocessing:

Data processing is among the most crucial steps in the complete machine learning project. The data is cleaned and processed to make it fit for learning and training the model. The cleaning process is very crucial as the data which is not clean can never give a good result even when tested on a perfect machine learning model. But well, clean and wrangled data, even when trained and tested on an average model, will give a decent result. So, the authors cleaned the data using BeautifulSoup library, and by using the bag of words approach and count vectorizer, they made the data clean and ready to be trained.

iii. Feature Selection:

Bag of word (BoW) is an approach for most of the NLP and ML techniques. Apart from BOW, the authors have used Word2Vec (both CBOW and Skip n-gram) along with this Doc2Vec have been tried in which the authors implemented PV-DM and PV-DBOW.

iv. Model Building:

For this project, the authors trained ml models using Linear Support Vector Machine Classifier, Random Forest Classifier, word2vec, and doc2vec. These machine learning algorithms are some of the most powerful approaches to work with Natural Language Processing problems.

B. IMPLEMENTATION

i. Bag of Words:

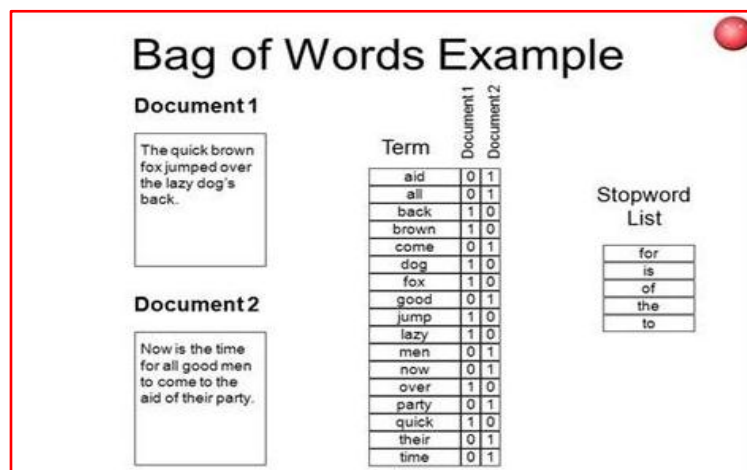
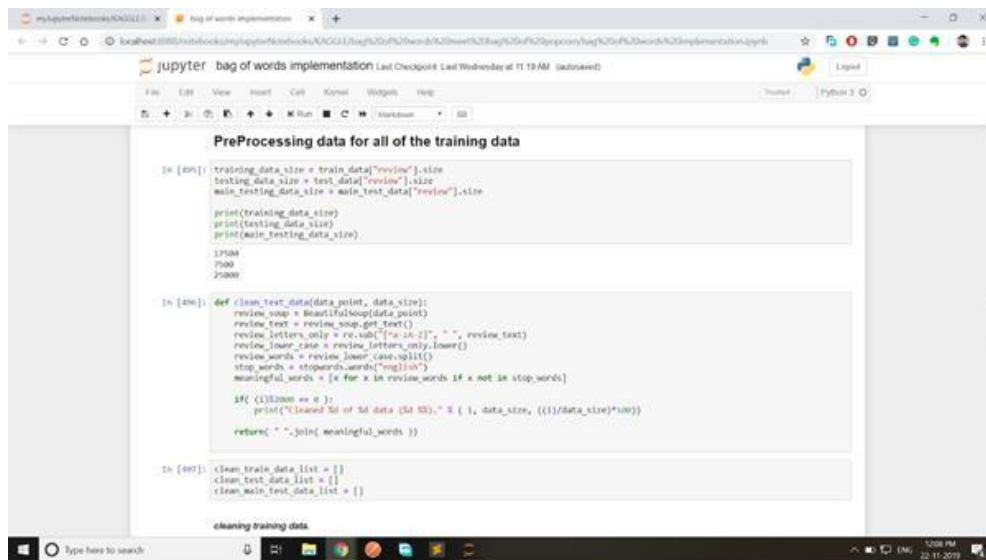


Fig. 1 Bag of Words

This model is a method for speaking to content information when displaying content with AI calculations. It is easy to comprehend and execute and has seen incredible achievement in issues, for example, language displaying and archive characterization and classifications. The main source of data which the authors used was from Andrew Maas, from Stanford University. The dataset consists of binary data which is comprises of 25,000 data of movie reviews which is for training and another 25,000 of the data for testing out machine learning model.

Bag of Words implementation can be seen in the Fig. 2.



```

In [495]: training_data_size = train_data["review"].size
testing_data_size = test_data["review"].size
main_testing_data_size = main_test_data["review"].size

print(training_data_size)
print(testing_data_size)
print(main_testing_data_size)

17500
2500
25000

In [496]: def clean_text_data(data_point, data_size):
review_soup = BeautifulSoup(data_point)
review_text = review_soup.get_text()
review_letters_only = re.sub("[^a-z]", "", review_text)
review_lower_case = review_letters_only.lower()
review_words = review_lower_case.split()
stop_words = stopwords.words("english")
meaningful_words = [x for x in review_words if x not in stop_words]

if (len(meaningful_words) > 0):
print("%cleaned %d of %d data (%d %%) " % (1, data_size, ((1)/data_size)*100))
return " ".join(meaningful_words)

In [497]: clean_train_data_list = []
clean_test_data_list = []
clean_main_test_data_list = []

cleaning training data.

```

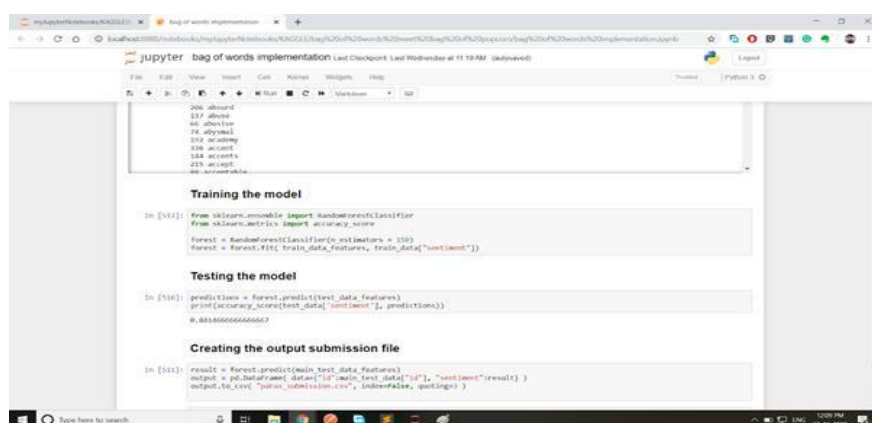
Fig. 2 Bag of Words Implementation

The above Fig. 2. shows the preprocessing of the data extracted from the dataset the authors collected above. The training and testing set keeps the feature of “review” which has the text data of the user's tweet reviews about a movie. Then the authors cleaned the data by creating an iterative loop of calling the function clean_text_data which do the following things: Called the BeautifulSoup library which removes all the HTML or XML tags from the data and makes it a non-coded text data.

Then using a regular expression, the authors removed the unwanted special characters from the text which are not useful for this analysis. Later, the authors made the text to be all lowercase to avoid any kind of case sensitive conflicts. Then they imported the stop words which do not give any sentiment in the corpus and are of no use and should be removed.

The final outcome of this process was clean and ready to be trained reviews of the users [6].

III. RESULTS AND DISCUSSIONS



```

200 absurd
137 above
60 abusive
14 abused
152 academy
188 account
144 account
225 accept
66 acceptable

Training the model

In [511]: from sklearn.metrics import RandomForestClassifier
from sklearn.metrics import accuracy_score
forest = RandomForestClassifier(n_estimators = 100)
forest = forest.fit(train_data_features, train_data["sentiment"])

Testing the model

In [512]: predictions = forest.predict(test_data_features)
print(accuracy_score(test_data["sentiment"], predictions))
0.8486666666666667

Creating the output submission file

In [513]: result = forest.predict(main_test_data_features)
output = pd.DataFrame({'id':main_test_data["id"], "sentiment":result})
output.to_csv("submission.csv", index=False, encoding='utf-8')

```

Fig. 3 Model Training and Testing

The below image Fig. 3. shows the training and the testing of the data on the main model which was a Random Forest Classifier.

Random Forest is an ensemble learning technique in which data is divided into multiple decision trees and passed to get divided into each node pass. The data then falls in one of the leaf nodes which defines its class. Random forest is used for both regression and classification type of learning algorithms. The Random forest algorithmic model for training the data extracted was imported from the scikit-learn library which has a collection of enormous number of machine learning algorithms. It was set to run on 100 decision trees and the result to be aggregated. Training data is fed in the classifier and the model is trained over that data.

Then accuracy score was imported to get the accuracy value for the prediction created. Prediction is calculated for the testing data input and then tested for the accuracy score with comparing with the already present output of the testing set. The accuracy comes out to be 84.14%.

Secondly, the authors have used ANN model as shown in the Fig. 4. with six layers as a classifier and they have obtained a remarkable 91.75% accuracy in training the model, but the performance went down to 49.90% in the testing set.

```
ANN Model Training
Let's go.

In [31]: EPOCHS = 10
         BATCHSIZE = 1500

In [32]: #submit log reports to tensorflow by specifying a path
         # SAVE MODEL FOR 5 EPOCHS
         mc = keras.callbacks.ModelCheckpoint('weights{epoch:08d}.h5', save_weights_only=True, period=5)
         tensorboard = TensorBoard(log_dir="logs/{}".format(time()))
         model.fit(trainX, trainY, epochs=EPOCHS, batch_size=BATCHSIZE, validation_split=0.2, verbose=1)

WARNING:tensorflow:From C:\Users\REVANTH\Anaconda3\lib\site-packages\tensorflow\python\ops\math_ops.py:306
6: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 20000 samples, validate on 5000 samples
Epoch 1/10
20000/20000 [=====] - 12s 580us/step - loss: 0.7008 - accuracy: 0.5121 - val_loss: 0.6780 - val_accuracy: 0.7222
Epoch 2/10
```

Fig. 4 ANN Model Training

Finally, the authors implemented Support Vector Machine Classifier as a training model for the same dataset which can be seen in Fig. 5. For the SVC model the authors got an accuracy 83.40%.

```
In [22]: #training the linear svm
         svm=SGDClassifier(loss='hinge', n_iter=500, random_state=42)
         #fitting the svm for bag of words
         svm_bow=svm.fit(cv_train_reviews, train_sentiments)
         print(svm_bow)
         #fitting the svm for tfidf features
         svm_tfidf=svm.fit(tv_train_reviews, train_sentiments)
         print(svm_tfidf)

SGDClassifier(alpha=0.0001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
              n_iter=500, n_iter_no_change=5, n_jobs=None, penalty='l2',
              power_t=0.5, random_state=42, shuffle=True, tol=None,
              validation_fraction=0.1, verbose=0, warm_start=False)
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
              n_iter=500, n_iter_no_change=5, n_jobs=None, penalty='l2',
              power_t=0.5, random_state=42, shuffle=True, tol=None,
              validation_fraction=0.1, verbose=0, warm_start=False)
```

Fig. 5 SVM Model

IV. CONCLUSION

The authors have used Bag of Words for feature vectors and they have implemented three different classification approach. The authors have observed the best one among the three models. The first model random forest gave us an accuracy of 84.14. The second model was trained using ANN (Artificial Neural Network) and obtained an accuracy of



49.90. Finally, the last model is trained using SVM (Support Vector Machine) which gave an accuracy of 83.40. As comprehended from the above conclusion, the authors observed that the Random Forest model gave us the highest accuracy of 84.14. The comparison among the three models based on accuracy is described in the Table 1. in tabular format and pictorially depicted in Fig. 6.

Table I Comparison Among the Three Models Based On Accuracy

Classification Model	Accuracy
Random Forest	84.14
Artificial Neural Network (ANN)	91.75 in training the model 49.90 in the testing set
Support Vector Machine (SVM)	83.40

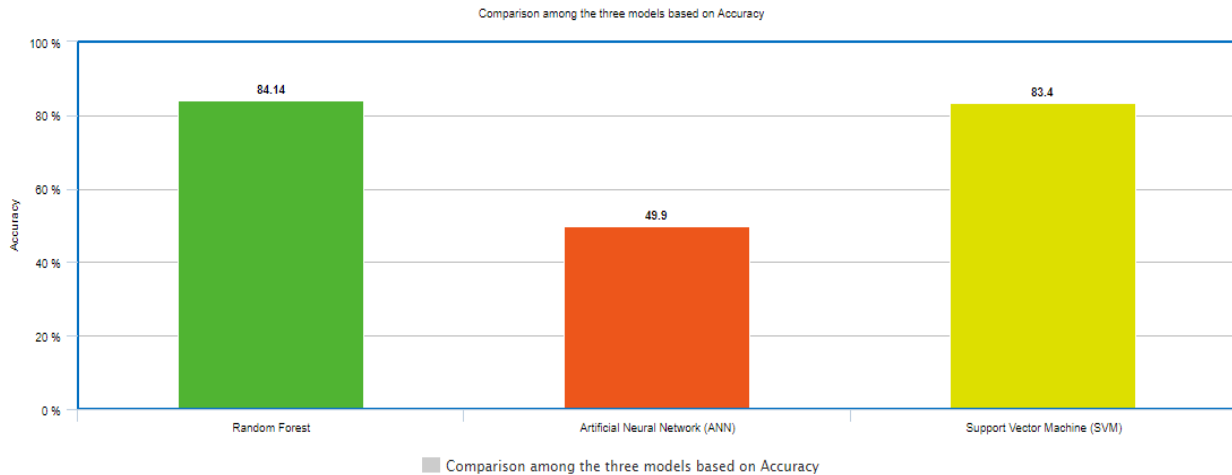


Fig. 6 Comparison among the three models based on Accuracy

ACKNOWLEDGMENT

The team members express their gratitude to **Dr. K.S. Geetha** and **Dr. K.N. Subramanya** for their continuous support and encouragement.

REFERENCES

- [1]. Maas, A L., Daly, R E., Pham, Peter T., Huang, D, Ng, Andrew Y., and Potts, Christopher. Learning word vectors for sentiment analysis, In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, Vol 1, Pg142-150, 2011.
- [2]. Kouloumpis, Efthymios, Wilson, Theresa & Moore, Johanna. Twitter Sentiment Analysis: The Good the Bad and the OMG!, International Conference on Weblogs and Social Media (2011).
- [3]. T Singh, M Kumari, Role of text pre-processing in twitter sentiment analysis, Procedia Computer Science Vol 89, Pg:549-554, 2016.
- [4]. Pang B and Lillian L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Proceedings of the Association for Computational Linguistics, pg: 115– 124, 2005.
- [5]. Socher, Richard, Pennington, Jeffrey, Huang, Eric H, Ng, Andrew Y, and Manning, Christopher D. Semi supervised recursive autoencoders for predicting sentiment distributions. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Pg:151-161, (2011c).
- [6]. Rafael Macheal Karampatsis, John Pavlopoulos and Prodromos Malakasiotis, Two Stage Sentiment Analysis of Social Network Messages, SemEval, Pg:114–118, (2014).
- [7]. Wang, Sida and Manning, Chris D. Baselines and bigrams: Simple, good sentiment and text classification. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Vol:2, Pg:90-94, 2012.