Machine and Deep Learning Classification for Sentiment Analysis On Food Reviews

Kushagra Gupta
190649351
Dr Thomas Roelleke
Msc. Big Data Science with IE

***Abstract*—Natural Language Processing (NLP) is gaining popularity in recent times due to development of Internet which changed the way people buy products online and share their reviews. Sentiment classification or opinion mining is one of the popular and challenging tasks of NLP [1] that recognizes polarity (e.g. a positive or negative opinion) within sentence, text, paragraph etc. Examining the consumers opinions helps business to determine the people point of view about a product and how it is received in the market. In this paper, the aim is to tackle the conundrum of sentiment polarity categorization, which is one of the fundamental problems in sentiment analysis. The online product review dataset used is from Amazon, which is a well-liked online food buying space. This work presents use of Vectorization techniques with Statistical language models like N-grams in NLP to generate features for Machine Learning models. This project intends to employ feature extraction technique like Bag of Words (BoW) and TF-IDF and have the different models like linear and non-linear classifiers such as Logistic regression, Decision trees and random forest learn on these features. And due to recent development of neural networks for sentiment analysis, the work also explores deep learning model like CNN. The experimental results demonstrate the proposed N-grams Statistical language model like Bigrams is capable to increase the accuracy of the proposed models. Results also provides a clearer view on how the CNN model using word-embedding provides better classification accuracy during sentiment analysis.**

Keywords—Sentiment Analysis, Machine Learning, NLP, N-grams, Linear vs Non-Linear, Deep Learning, CNN.

I. Introduction

The Internet has now made us slave in our day to day lives and we are totally dependent on it on everything even for the smallest bits of our lives. People use internet for social media, shopping online or ordering food. There are tons of food delivering apps like Deliveroo, uber eats, grocery shopping apps like Aamazon, Tesco and food reviews websites like TripAdvisor, yelp. where people share their reviews. All these platforms provide a way to the customer/reviewer to write his/her comments about the service or product and give a rating for it. Online consumer reviews are more user-oriented and describe the products from the user's perspective.

"What consumers might be thinking ?" is the usual question asked by business around the globe in decision-making process. Based on these comments and ratings Business firms are now making their products and services better. Using sentiment analysis to do predictive analysis of reviews about a product could bring to light if customers are satisfied about pricing plans and customer service. For most of the companies, the revolutions in social media is far-reaching and comprehensive, given the global scale on which people are interacting and the amount of online discussions, chat, dialogue taking place, a huge chunk of which possess the potential to yield new consumer insights. As more companies are realizing the advent of the internet and consumer sharing their opinions, these consumer interaction can help to shape the opinions of other consumers, ultimately their brand fidelity and their own brand endorsement [2].

In this work, we try to experiment and analyze on one such popular food space Amazon.com, which is the top choice amongst people around the globe for online products. Figure 1 shows how the number of customer reviews increased from 2005-2012 multifold for food.

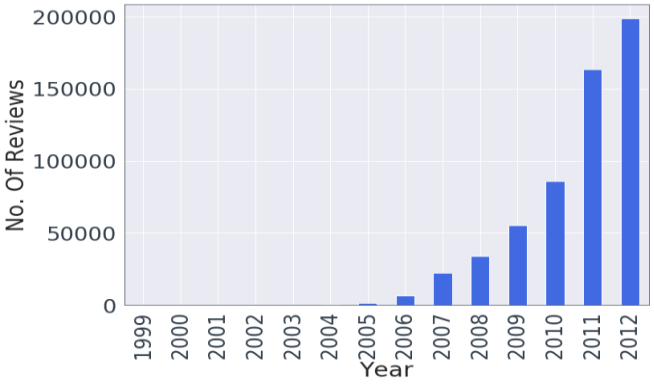

Figure 1: Number of reviews per year from Amazon Dataset

The work focuses on to build classical machine learning models like logistic regression by keeping major attention to choose the right vectorization technique and N-grams Statistical language model for classification task. The models are built for two classification tasks, a binary 2 class sentiment (positive and negative) and 5 class sentiment (very poor, poor, average, Good, very good). The work experiment to how different vectorization techniques can help to increase the accuracy in the initial stages. We also experiment comparisons between different N-grams model like unigram, bigrams, trigrams etc. The Bigram model works well and shows accuracy of well above 93%. Furthermore, we also show comparisons between 2 class vs 5 class model with different vectorization techniques as an input to logistic regression. Comparing different linear and non-linear models, work concludes that the classical linear logistic models provides better results than non-linear tree model and performs roughly comparable to the Deep learning CNN model.

II. Background Research and Literature Review

A lot of research has been done in this field for more than 2 decades now and has been growing exponentially. "Why opinions are important?" The answer to this question was provided by the ComScore study [3] conducted from October 12-18, 2007 was done on 2,078 people, including 508 who used online platform for reviews. Consumers were questioned how they use online reviews and the price they would be willing to pay for a service given an average service price and an aggregate consumer rating. The study examined

the offline sales impact of online reviews for restaurants, travel, automotive, hotels, medical and home services. The results were quite significant as per sentiment analysis perspective [4].

- On a typical day, 15% of Americans shared their opinion for any product.
- 81% of users who uses Internet, did online research on a product at least once before purchase.
- More than 79% of user found reviews having a meaningful impact on their buying habits.
- Consumers were willing to pay a more than 20% for a product that has been rated more than 4.

A. *Opinion mining and sentiment classification*

The previous works done on this topic have focused on polarity categorization and use of machine learning for sentiment classification. Pang, B Lee [12] studied the problem of classifying documents by overall sentiment, categorizing review as positive or negative. Using movie reviews dataset, they used customary machine learning techniques which outperformed human-produced baselines. Although, the machine learning methods used (Naive Bayes, maximum entropy classification, and support vector machines) do not perform as well on sentiment classification as on traditional topic-based categorization. Na JC et al [13] presented study in sentiment classification, categorizing online review text in relation to the overall sentiment expressed in them. The paper presented a blueprint of sentiment-based meta search engine that was reinforced to carry out sentiment categorization of web search results with the help of a self-regulating classifier which is established on a supervised machine learning algorithm.

Rather than focusing on a lot of machine learning techniques the major area of research in this work has been to choose the correct data which will be fed to most popular and widely used classifier logistic regression and analyze how the feature extraction techniques makes impact in improving accuracy of the model.

B. *Statistical language model (N-grams)*

N-grams for texts are profoundly used in opinion mining and natural language processing tasks. In this work we have taken significant use of N-grams to further improve accuracy of the models. The use of N-grams started way back when Gayo Avello et al [14] work was to test the application to IR of a modified version of the N-gram where vector space model relative frequencies were no longer used as vector weights but were replaced by N-gram significances. The new approach was successfully applied to other NLP tasks such as language identification or text summarization and the results were encouraging. Furthermore, Ye Qiang et al [15] researched on rapid growth in Internet applications in tourism and travel-related information. They used supervised machine learning algorithms along with N-gram model(character-based) for sentiment classification of the reviews on travel blogs for seven popular travel destinations in the US and Europe. The work incorporated sentiment classification techniques to the domain of mining reviews from travel blogs. Their results showed that N-gram approaches outperformed the state of arts approaches.

C. *Deep Learning for Sentiment Classification*

Convolutional neural networks (CNN) date back decades [7]. CNNs are used for comprehension in NLP and speech recognition, although frequently for NLP, Recurrent Neural Nets (RNNs) are used. The papers [8][9][10][11] showed use of CNN on sentence categorization to make use of 1D structure of text data for correct prediction. The CNN is applied to high-dimensional text data, instead of low-dimensional word vectors as is often done. Jiayu Wu, Tianshu Ji [5] studied the applications of Recursive Neural Network on sentiment analysis tasks to process the raw text data from Amazon Fine Food Reviews. They proposed a novel technique and implemented to parse binary trees using Stanford NLP Parser and due to lack of labelling in the original dataset they labelled tree nodes to achieve the level of supervision that RNN required. They proposed a new model RNNMS (Recursive Neural Network for Multiple Sentences) which showed better results than machine learning models.

A lot of work has shown use of deep learning for text classification and deep CNN's have shown explosive popularity. The use of Deep learning methods like LSTM and CNN are becoming more prominent in sentiment analysis. Zhenxiang Zhou and Lan Xu [6] work exhibited models to solve the review usefulness classification problem. Their works showed how both feed-forward neural network and LSTM were able to outdo traditional models and LSTM outperformed feed-forward neural network. Ishita Chakraborty, Minkyung Kim, and K. Sudhir [16] also focused on CNN-LSTM which take advantage of the spatial and sequential structure of language.

In this paper, we exploit the use of CNN which has not been researched much with the Amazon dataset, to provide the best performance in training data size requirements, training speed and accuracy.

III. Research Design and Methodology

The design used to create the framework for this study has been shown in the Figure 2. The research focuses on both qualitative as well as quantitative methods during the model design and analysis of data. Different phases of development are discussed to show use of different Artificial Intelligence techniques for text classification.

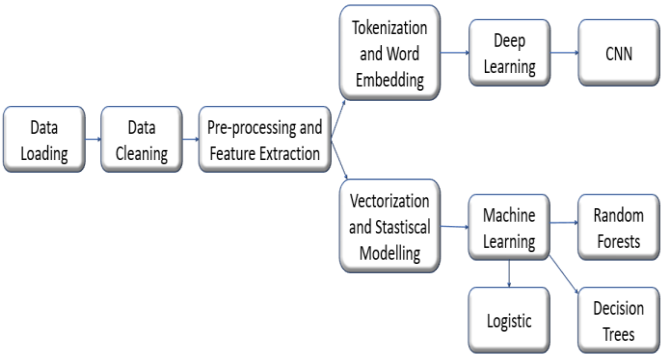

Figure 2:Design

A. *Data Collection*

The Data set used is Amazon food reviews dataset [17] which has been collected from amazon.com. The dataset has 568454 review rows provided by user between October 1999

to October 2012 which comprises of 2.8 million sentences, containing more than 45 million words and with an average five sentences per review.

Figure 3 shows dataset sample, two main fields which are used for the analysis are the ones which contain predictive value:

Score: Rating between 1 and 5 for a specific product is stored in this column.
Text: The review for a specific product by a specific user is stored in this column.

Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
3	B000LQOCH0	ABXLMWJXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...

Figure 3: Dataset Sample

B. Data Cleaning and Preprocessing

The data extracted from the Text field needs processing to remove all the non-textual information like special characters, dropping all the null fields, stop words and text normalization.

Stop Words: These are the most mundane words such as ‘is’, ‘on’, ‘the’, etc which add little or no value to the document and provide no helpful information used for classification. These take database space and valuable processing time. Natural Language Toolkit (NLTK) has been used to clean the Text field.

Lemmatization: This is a very crucial preprocessing technique of text normalization to transform word into canonical form to reduce the number of unique words and reduce the training time of the model. This technique makes use of vocabulary along with word structure (morphological Analysis) and create detailed dictionaries from all conjugation forms of words known as lemma. For example, the words ‘studies’ or ‘studying’ will take similar lemma i.e. ‘study’.

C. Exploratory Data Analysis

The Score field has value between 1 and 5 which has been changed to two different types of multiclass classification.
2-Class Classification: In this type of classification the score value greater than 3 has been considered as ‘positive’ sentiment and equal or less than that as ‘negative’ sentiment. Figure 4 shows how the class labels looks like.

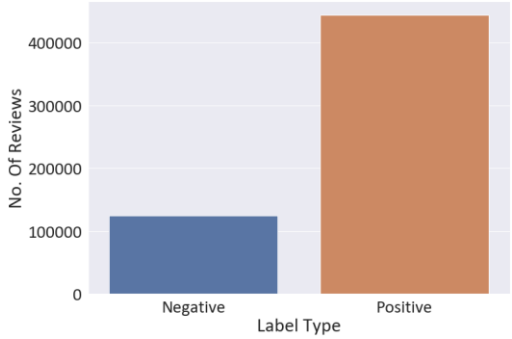

Figure 4: 2 Class Classification

The mean of score is well over 4 and the distribution has more positive than negative reviews.

5-Class Classification: The score values here are divided as per the score, the lowest score (1) has been labelled ‘very poor’ and highest score (2) as ‘Very good’. Figure 5 shows the number of reviews for this classification.

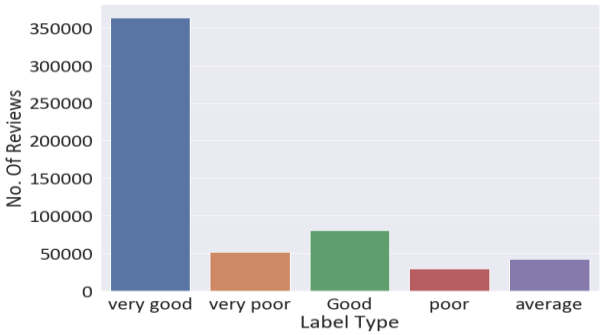

Figure 5: 5 Class Classification

After the text pre-processing it is always essential to visualize how the people use positive and negative words for reviewing any product. Figure 6 and Figure 7 shows the frequency of words in the reviews with the help of word cloud.
The Positive word cloud shows the occurrence of adjectives like ‘good’, ‘great’, etc., also the data shows candy, oatmeal, taffy was some of the products frequently bought and reviewed positively by people.

Figure 6:Positive Words in the reviews

In the Negative word Cloud, flavor, size, dog food seems to be major problems among users in the reviews. Words like ‘not good’, ‘disappointed’ were frequently used in the negative reviews.

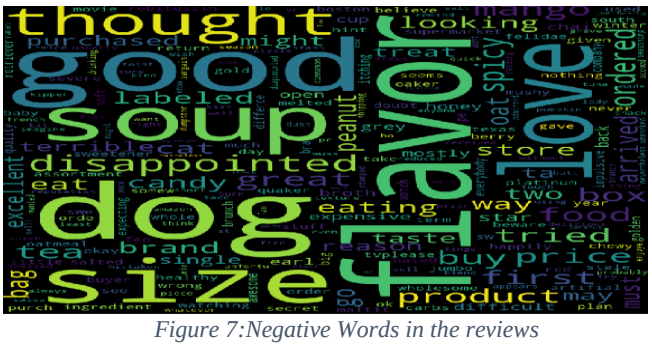

Figure 7:Negative Words in the reviews

D. Feature Extraction

Extracting features from text documents is an essential part as raw data cannot be fed directly to the algorithms. In

the experimental process numerical features vectors are extracted using different vectorization techniques given as below which will be fed further in different machine learning and deep learning models.

a) Bag-of-Words

It is an Information Retrieval technique used widely to create feature vectors from text documents. BoW creates vocabulary of the unique words where each index corresponds to one word and each word is a different dimension. BoW creates the matrix of all the words and fills the matrix with raw count of tokens in each document, which is known as the **term frequency (TF)** given by.

$$tf_{t,d} = f_{t,d} \quad (1)$$

where f is the raw count, d is document and t is token.

Considering our dataset text if we have N sentences {S1, S2, ...SN} and T unique tokens on total.

S1: "The dog food was excellent"

S2: "The Food has excellent flavor"

The BoW will create the vocabulary for unique tokens ['The', 'Dog', 'food', 'was', 'Excellent', 'flavor', 'has'] Then will create matrix from this vocabulary of size NxT, given as below.

	The	Dog	food	was	Excellent	flavor	has
S1	1	1	1	1	1	0	1
S2	1	0	1	0	1	1	0

The above matrix depicts the training features containing term frequencies of each word in each review. This is the most simplistic and easy to implement approach as only number of occurrences matters and not sequence or order of words.

b) Term Frequency Inverse Document Frequency (TF-IDF)

TF-IDF helps to look at a normalized count where each word count is divided by the number of documents this word appears rather than looking at the just the counts of each word in each document. The whole idea of TF-IDF is to less value the frequent terms in the document corpus while scaling up the rare ones. IDF or Inverse document frequency is the inverse of number of documents that a term appears or the inverse of document frequency by compensating the rarity problem in BoW model. TF-IDF uses **term-frequency (TF)** as defined in BoW times **inverse document-frequency (IDF)** given by,

$$tfidf_{t,d} = f_{t,d} \times invf_{t,d} \quad (2)$$

In this work the TF-IDF is used from scikit-learn library [18] and is computed as given below.

$$tfidf_{t,d} = log\left(\frac{1+n}{1+df(t)}\right) + 1 \quad (3)$$

where ***n*** is the total number of documents in the document set, and ***df(t)*** is the number of documents in the document set that contain term. The resulting TF-IDF vectors are then normalized by the Euclidean norm [18] given by,

$$v_{norm} = \frac{v}{\sqrt{v1^2+v2^2+\cdots+vn^2}} \quad (4)$$

The TF-IDF value grows proportionally to the occurrences of the word in the TF, but the effect is balanced by the occurrences of the word in every other document (IDF).

c) N-grams

The N-grams are very useful when developing language models. When the individual words are considered their might be chances where things can go wrong with predictions for instance if review has sentence "food is not good" and we consider individual words, the review can be categorized both as negative("not") and positive("good") sentiment. But if we consider the bigram model where two words are taken together in consideration then the sentiment would be classified correctly. N-grams are group of co-occurring words within a defined slot which may be a unigram (1 word), bigram (2 words together), trigram (3 words together) and so on. The N-grams is given by,

$$\text{Ngrams}_R = X - (N-1) \quad (5)$$

Where X is number of words in each review R.

d) Tokenization and sequence padding

Since neural networks work by performing computation on numbers and passing in a raw word will not work. Tokenization coverts text sentences to numeric representation mapping. The **fit_on_text** step, depending on word frequency creates the vocabulary index. For example, if input sentence is 'Very good', then the dictionary created will be word_index['Very'] =1 and word_index['good'] =2. Every word gets unique integer value. Now that the data has been tokenized and have a word to numeric representation mapping of our vocabulary, we use it to encode our sequences. The **text_to_sequence** step considers every word in the document and substitutes it with its corresponding integer value present in dictionary of word_index.

Now that the sequence has been created there will be cases where the sequences are not of equal lengths. Padding process helps to make the sequences equal by adding '0' to those which are shorter.

e) Word Embedding

It is an NLP technique intended to map semantic significance to a geometric space and is used at the input stage of a neural network. In this work, these words embedding have been fed in the CNN model as the first input layer of the model. The process starts with aligning a numeric vector to every word in a dictionary, such that the distance between any two vectors would capture part of the semantic association between the two associated words. The geometric space formed by these vectors is called an *embedding space* [19].The distance commonly used in the process is L2 or cosine distance. For understanding this technique, the words "dog" and "bark" have some relation and their representation is embedded closely. But if we consider two words "football" and "fruit" which are different semantically so vectors representation in embedding space will be very far apart.

IV. MODELS

A. Logistic Regression

Logistic regression as all other regression is a predictive analysis which optimizes a discriminative objective function. It is used to bring out relationship between one dependent and one or more set of ordinal independent variables. The Logistic regression also known as logarithmic logit regression is used in classification tasks because the dependent variable is categorical. The Question arises here is

"Is logistic regression linear?", the answer is yes, and we know behind every great Logistic Regression model is an unobservable linear regression model, because the question logistic is trying to answer is:
"What is the probability an observation belongs to class 1 given some characteristics x?"

$$p(y = 1|x) \quad (6)$$

To calculate this probability, we take the integral of the logistic distribution to get its cumulative distribution function:

$$p(y = 1|x) = \sigma(\theta^T x) = \frac{1}{1+\exp(-(\theta^T x))} \quad (7)$$

which is nothing but the sigmoid function, which helps to map predicted values to probabilities. The factor making logistic linear is the decision boundary, given by

$$\theta^T x = 0 \quad (8)$$

Logistic regression divides the feature space X with a hyperplane which is the property of a linearity.
Logistic algorithm used in the experiment is from scikit-learn library [20]. L2 regularization has been used to improve numerical stability. The cost function used is as given below.

$$J(\theta) = \min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log\left(\exp\left(-y_i\left(X_i^T w + c\right)\right) + 1\right) \quad (9)$$

The algorithm to use for optimization, in scikit-learn terms solver used is 'sag' which uses Stochastic Average Gradient descent [21]. It works very well when the dataset is large.

B. Decision Trees

It is a non-linear supervised tree method-based machine learning algorithm which provide good accuracy and stability at solving both classification and regression problems. A decision tree is a waterfall structure in which each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label [28]. The paths created from root to leaf represent classification rules.

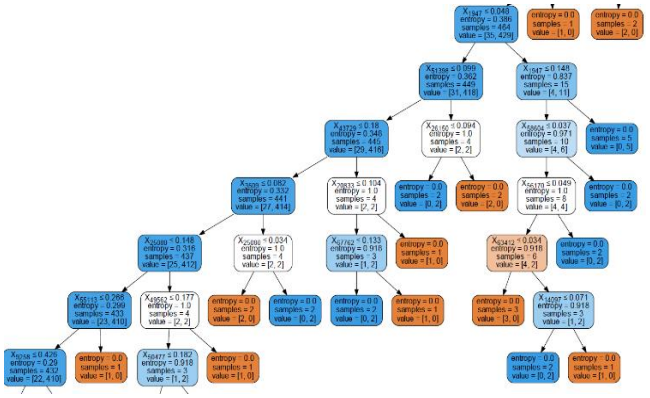

Figure 8:Sample Decision tree from the experimental result

The decision tree version used here is from scikit-learn [23]. which uses CART (Classification and regression Trees) algorithm. This algorithm creates a binary tree by finding the best categorical feature to split using impurity criterion. The criterion used here is called Gini given by,

$$Gini = \sum_{i=1}^{C} f_i(1 - f_i) \quad (10)$$

where *fi* is frequency of the label *i* at a node and *C* is number of unique labels.

C. Random Forest

Random forest is a type of ensemble tree-based learning algorithm. It is a non-linear model used for both classification and regression task. It uses ensemble algorithm which combines more than one algorithm of same or different kind for classifying objects. At training, many individual decision trees are created, and the prediction results are summed up to make the final prediction.
The random forest classifier used is from scikit-learn library [22]. When trees are built by the algorithm, the decision about which variable to split at each node uses a calculation of the Gini impurity given by,

$$Gini = 1 - \sum_{i=1}^{C} (f_i)^2 \quad (11)$$

where *fi* is relative frequency of the class and *C* represents the number of unique labels.
In this algorithm node importance is calculated for each decision trees given by,

$$ni_j = \sum_{j=1}^{n} w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)} \quad (12)$$

where *ni* is the importance of node j, w is w weighted number of samples reaching node, C is the impurity value, left(j) and right(j) are child node from left split and right split.
Importance of feature is then calculated given by,

$$fi_j = \frac{\sum_{j:node\ j\ split\ on\ feature\ i} ni_j}{\sum_{k \in all\ nodes} ni_k} \quad (13)$$

where *fi* is importance of feature i, ni is importance of node j.
Normalization is done for these feature importance value,

$$normfi_i = \frac{fi_i}{\sum_{j \in all\ features} fi_j} \quad (14)$$

And at last the sum of the feature's importance value on each tree is calculated and divided by the total number of trees:

$$RFfi_i = \frac{\sum_{j \in all\ trees} normfi_{ij}}{T} \quad (15)$$

Where $RFfi_i$ is the importance of feature *i* calculated from all trees in the Random Forest model, T is total number of trees.

D. Covolution Neural Network (CNN)

The advancements in deep learning has provided new standards to evaluate sentiment analysis models and has given a lot of customary model architectures that can be swiftly revamped to any dataset to achieve high accuracy. Convolutional Neural Networks have been providing breakthrough in image classification but most recently are now being used in solving NLP problems as the proposition is that instead of carrying out convolutions on image pixels, the model can implement convolutions in the embedded feature space of the words in textual data.
"So exactly what are convolutions?", It can be understood by simply thinking of it as, if a window(filter/kernel) of some size *NxM* has been created and slided all over the input matrix, multiplying the values element-wise during each step movement and summing up the result to create a new matrix. CNN's are multiple layers of these convolution with non-linear activation functions (ReLU). The input provided to these networks are sentences represented as a matrix, where each row of this matrix represents one word or here as token. These tokens are nothing but vectors representing a word

created using preprocessing technique like word embedding which has been used in this work. Figure 9 is depiction of the architecture used in this work.

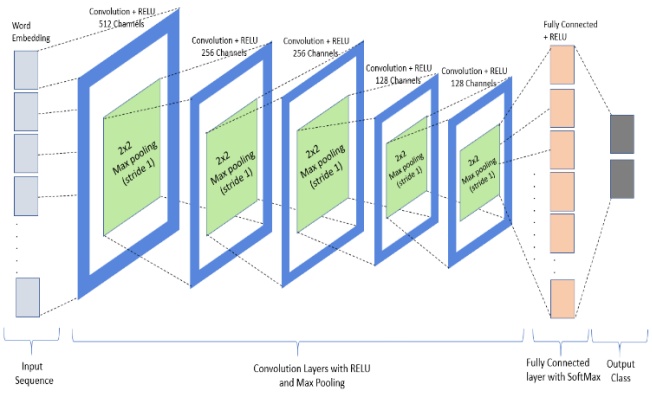

Figure 9:CNN architecture

Some of the important functions used in this work to build the model are briefly explained.

ReLu: It is a type of activation function which is used to increase the non-linearity of the input given by,

$$y = max\,(0, x)\; R(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \qquad (16)$$

which means it is linear for all positive values and zero for all negative values.

Max pooling: Pooling helps to down sample feature maps by summarizing the feature in the feature maps. The model in this work has been trained on two pool sizes 1(One word considered) and 2(Two words considered). The kernel of defined size is used to extract the feature which convolve around the matrix.

SoftMax: SoftMax is an activation function like ReLU used on the output of the very last layer. It is useful in multiclass classification because SoftMax gives us a discrete probability distribution over all the classes. It is given by,

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \qquad (17)$$

where N is number of classes, z is input vector, and $\sigma(z)_i$ is output class probability.

V. Experimental Results

A. Evaluation Methods:

a) Accuracy Score

Accuracy helps to measure the closeness of the predictions. For multilabel classification, the function returns the subset accuracy. If y_{pred} is the predicted value of the i-th sample and y_{actual} is the corresponding true value, then prediction accuracy is given by [25],

$$accuracy = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1\,(y_{pred} = y_{actual}) \qquad (18)$$

b) F-1 Score

Performance of each classification model is estimated base on its F1-score. The F1 score is defined as the harmonic mean of recall and precision given by,

$$F1\ Score = 2 * \left(\frac{precison * recall}{precision + recall}\right) \qquad (19)$$

c) Log Loss

Log loss also known as cross-entropy loss is used for estimates in logistic regression. For binary classification with a true label $y \in \{0,1\}$, and a probability estimate $p = \Pr(y = 1)$ the log loss per sample is the negative log-likelihood of the classifier given the true label is defined by[26],

$$L_{\log(y,p)} = -(\,y\,log(\mathrm{p}) + (1 - y)\;log(1 - p)) \qquad (20)$$

d) ROC Curve and AUC

Receiver operating characteristic or simply ROC curve, illustrates the performance of a classifier system using a graphical plot. The plot is created by taking the fraction of true positives out of the positives (True positive rate) vs. the fraction of false positives out of the negatives (False positive rate), at various threshold settings. AUC is defined as area under the receiver operating characteristics which helps to identify and compare ROC curve of other models.

B. Choosing the right vectorization:

The preprocessed data for two classification, 2-class and 5-class is fed to logistic regression. The data for both the types are converted using vectorization technique BoW and TF-IDF. Figure 10 shows comparison chart of logistic model trained on two different vectorization methods.

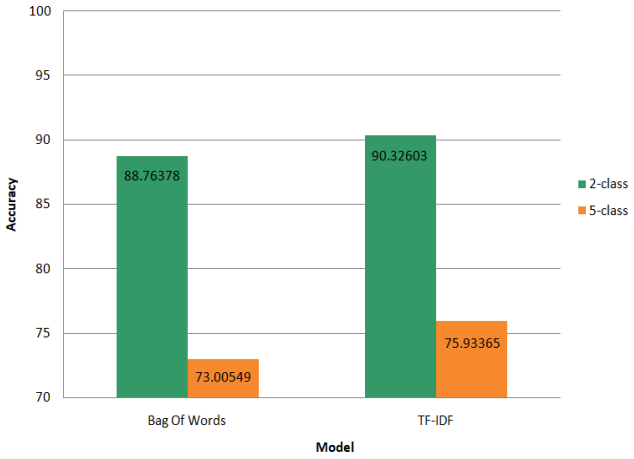

Figure 10:Vectorization Comparison results

The results show TF-IDF outperforms BoW in this classification task. The 2-class data was easy to predict and logistic regression performs well reaching accuracy above 90%. The 5-class data provides accuracy well over 70%, which is good as the review Score mean was around 4 which indicates less data to train for Score from 1-3.If the scores would have been evenly spread then the 5-class classification would be of great use but for now we stick with the 2-class('Positive' and 'Negative') sentiment to evaluate our models.

C. Choosing the right N-gram model:

As TF-IDF works well with 2-class data we further use this vectorization technique to train our logistic model by using N-gram method. In the experiment unigram (1 word), Bigram (2 words) and trigram (3 words) were taken into consideration. Figure 11 gives how the different N-grams model performed with logistic regression.

The results were encouraging as Bigram model provided excellent result, providing 3% more accuracy than the

unigram model which indicates how essential is word consideration in NLP. The Trigram model performed well as Bigram but at the cost of computation time.

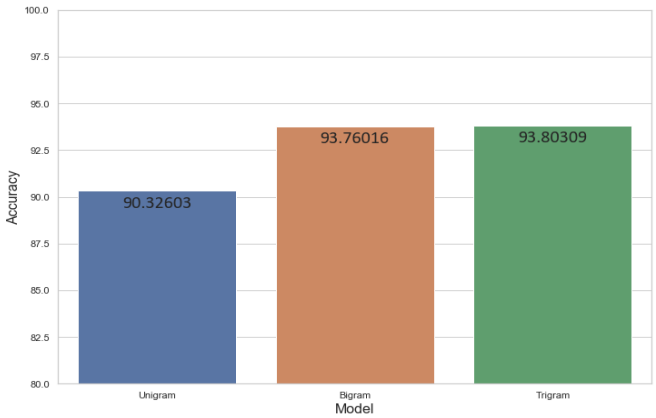

Figure 11: N-gram Model Comparison

Table 1: N-grams Evaluation results

Model	Vectorizer	Accuracy	log-loss	Precision	Recall	F1-Score
Unigram	TF-IDF	90.326031	0.243076	92.294233	95.586830	93.911680
Bigram	TF-IDF	93.760160	0.168647	94.827238	97.314232	96.054640
Trigram	TF-IDF	93.803086	0.171585	94.961076	97.219568	96.077051

If we go beyond trigram then the computation time will increase with little or no improvement over the accuracy which will be costly for larger dataset.

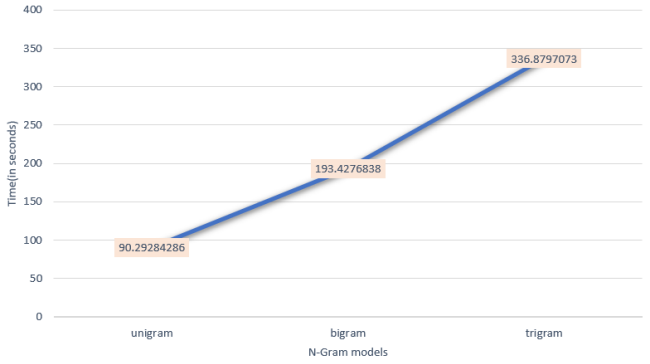

Figure 12: Computation cost for N-gram Models

D. Linear vs Non-Linear Performance:

After the N-gram has been selected we train the chosen linear and non-linear models with bigram vectorized data. The Non-linear model trained are decision tree and random forest which is later compared with the logistic model. Figure 13 depicts comparison bar graph for all the models. The result shows how the decision tree performs poorly as compared to other models due to their characteristics of always being prone to errors when solving classification problems and hence are computationally expensive than other models. Random forest provides accuracy of around 90% which is good, but they do not improve further even after tuning due to similar limitations, as it also considers multiple decision tress. Logistic model has performed excellently on all vector sets. Confusion matrix provides better picture on model performance categorization on reviews plotted for the model given by Figure 20,Figure 21 and Figure 22.

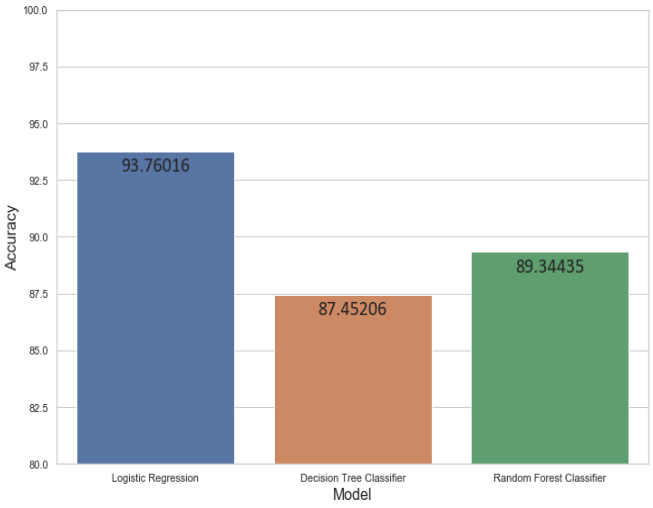

Figure 13: Linear vs Non-Linear Model Results

Table 2:Linear vs Non-Linear Evaluation results

Model	Vectorizer	Accuracy	log-loss	Precision	Recall	F1-Score
Logistic Regression	Bigram(TF-IDF)	93.760160	0.168647	94.827238	97.314232	96.054640
Decision Tree Classifier	Bigram(TF-IDF)	87.452059	4.332041	91.580828	92.420527	91.998762
Random Forest Classifier	Bigram(TF-IDF)	89.344349	0.487324	88.945365	98.603473	93.525740

ROC curve based on all model set is plotted in Figure 14.

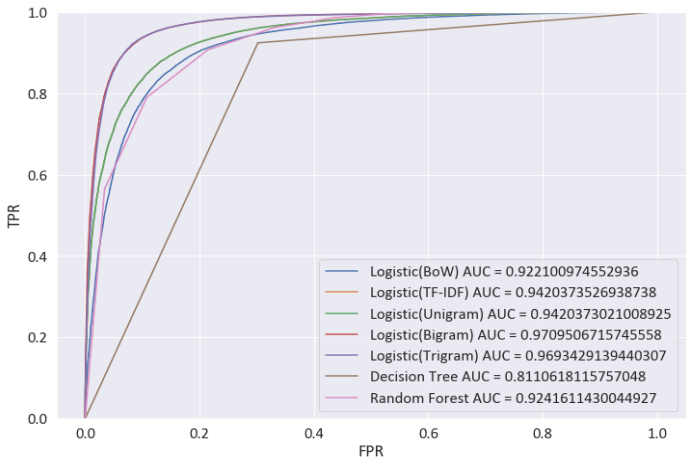

Figure 14:ROC and AUC for All models

E. CNN Model Performance:

The model (Figure 9) has been trained with 2-class data; data has been tokenized by creating sequences of vectors from text. The embedded sequence has been used as an input. Two models have been used considering one word and two words together. The models were trained on 10 epochs with batch size set to 64 and learning rate to 0.001 using Adam optimizer [27].

a) Considering One word

The pool size here is kept to 1 and stride at 1 to read one word at a time, working like a unigram model (Model Summary: Figure 23). The model training time is quite costly as no pre-trained weights were used but reaches accuracy above 91% and can be further increased using pre-trained model weights. Figure 15 and Figure 16 are resulting

accuracy plot and loss plot. The results indicate that accuracy was better than unigram logistic model.

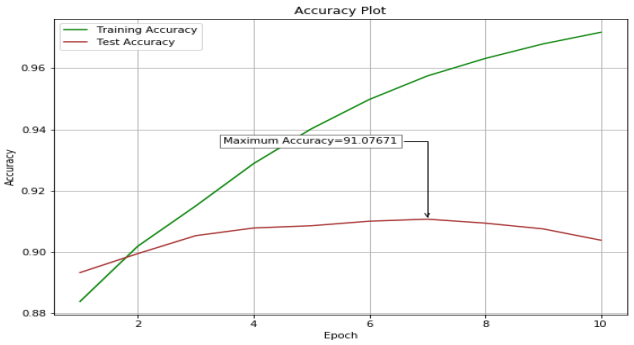

Figure 15:Accuracy plot (Considering One word)

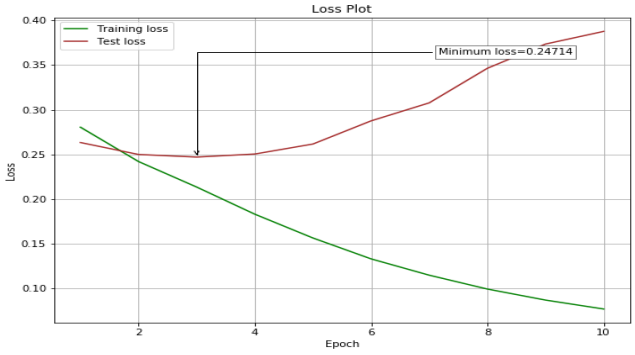

Figure 16:Loss plot (Considering One word)

b) Considering Two words

The pool size here is kept to 2 and stride at 1 to read two word at a time, working like a bigram model (Model Summary: Figure 24). The model training time was equivalent to one-word model, but accuracy was better with 91.57% and can be further increased using pre-trained model weights. Figure 17 and Figure 18 are resulting accuracy and loss plot.

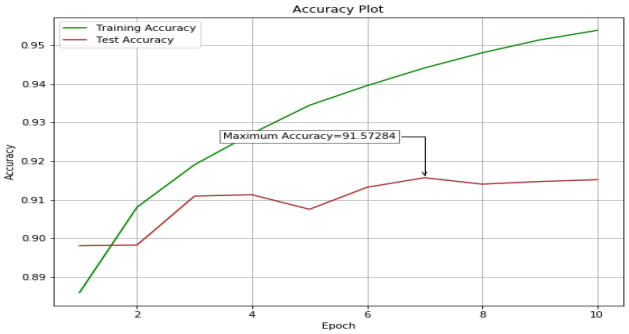

Figure 17: Accuracy plot (Considering Two Words)

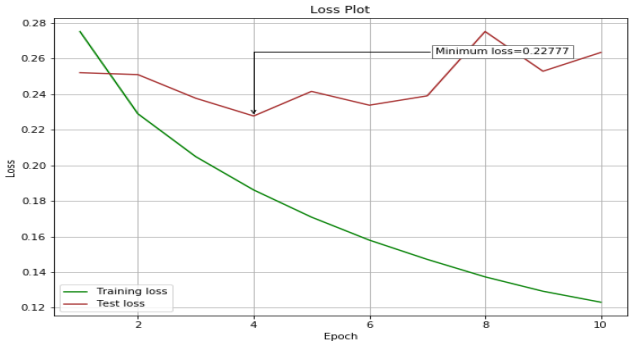

Figure 18:Loss plot (Considering Two words)

F. Model Comparison

Figure 19 shows comparison of all the models used in this study. The Classical machine learning algorithm logistic regression outperforms all the models due its robust nature to perform well with the classification dataset. We can see from the plot that Non-Linear models are comparable to other models but due to large structure to classify they become computationally expensive and show instability due to increasing complexity hence have bounded accuracy performance. The F1 score of 2-class model using bigram with the logistic model reaches well above 0.96 with AUC around 0.97, which is as accurate as prediction can be. CNN model works well in classifying the sentiments and is comparable with the logistic model both in term of accuracy and loss.

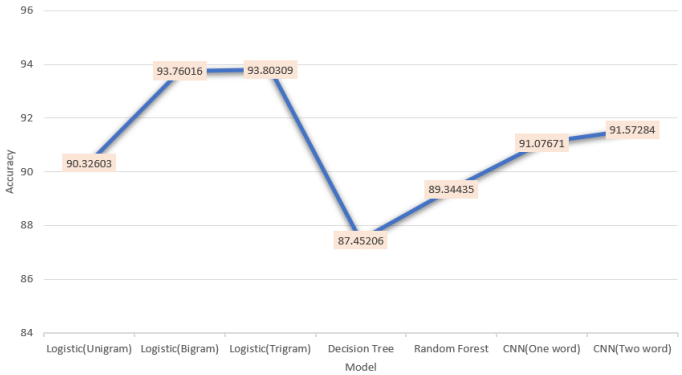

Figure 19: Model comparison

VI. Conclusion

The advent of web has led to increased customer interaction directly with the businesses and will be an important asset for sentiment analysis as future holds. This paper presents a novel approach to tackle the sentiment polarity categorization by choosing the correct vectorization techniques to train models. The experiments tried to conjecture that N-gram model can provide additional performance gain. Logistic bigram model has performed exceptionally well reaching accuracy of around 93.7 % and achieved superior performance over the state-of-the-art methods. Besides, proposed approach can be used to multiclass label categorization in sentiment analysis.

VII. Future Work

In this work major focus has been put in to classify 2-class sentiment (“Positive” and “Negative”) and sparsely on 5-class sentiment (Multiclass) due to sparse data present in score less than 3. This poses few limitations to this study as 5-class classification always helps to predict accurate and correct sentiment out of textual data. With this limitation in mind future work will be to focus on getting larger dataset with more data that is equally spread around review score and then exploiting the use of N-grams methods or CNN model with deeper layers to find the accurate results.

The use of Feature extraction technique like TF-IDF poses some limitation as they do not make use of semantic closeness within words and performs slow when data vocabulary is quite large. The future work will focus to implement new upcoming advanced feature extraction techniques like spacy, genism, fastText etc.

REFERENCES

[1] D. Osimo1 and F. Mureddu, “Research Challenge on Opinion Mining and Sentiment Analysis”.

[2] J. Zabin and A. Jefferies, “Social media monitoring and analysis: Generating consumer insightsfrom online conversation”, Aberdeen Group Benchmark Report, Jan 2008.

[3] The Kelsey Group, “Online consumer-generated reviews have significant impact on offlinepurchase behavior”. Press Release, November 2007.[Online]. Avaliable: https://www.comscore.com/Insights/Press-Releases/2007/11/Online-Consumer-Reviews-Impact-Offline-Purchasing-Behavior.

[4] B. Pang1and and L. Lee. (2008). Opinion mining and sentiment analysis.

[5] J. Wu and T. Ji. (2016). Deep Learning for Amazon Food Review Sentiment Analysis.

[6] Z. Zhou and L Xu. (2016). Amazon Food Review Classification using Deep Learning and Recommender System Stanford University.

[7] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel, “Backpropagation applied to handwrit-ten zip code recognition. Neural computation”, 1989, (pp. 541–551).

[8] C. Santos, N. Dos, and M. Gatti, “Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts”. In COLING, 2014 (pp. 69–78).

[9] Y. Kim, “Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing”, EMNLP, 2014, 1746–1751.

[10] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, “A Convolutional Neural Network for Modelling Sentences”, 2014. Acl, 655–665.

[11] R Johnson and T. Zhang, “Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. To Appear”, NAACL-2015, (2011).

[12] B Lee Pang, and S. Vaithyanathan. (2002). Thumbs up? Sentiment classification using machine learning techniques.

[13] JC Na, C. Khoo, and S. Chan. (2006). A sentiment-based meta search engine.

[14] G. Avello and A. Gutierrez. (2005). Application of variable length N-gram vectors to monolingual and bilingual information retrieval.

[15] Q. Ye, Z. Zhang, and R Law. (APR 2009). Sentiment classification of online reviews to travel destinations by supervised machine learning approaches.

[16] I. Chakraborty, M. Kim, and K. Sudhir, “Attribute Sentiment Scoring With Online Text Reviews : Accounting for Language Structure and Attribute Self-selection”, Yale university, May 2019.

[17] SNAP, “Amazon Fine Food Reviews”, May 2017, Distributed by kaggle, Avaliable: https://www.kaggle.com/snap/amazon-fine-food-reviews .

[18] Tf–idf term weighting, section:(6.2.3.4), Accessed: Aug.03, 2020.[Online]. Avaliable: https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction .

[19] Word embeddings in a keras model, Accessed: Aug.03, 2020.[Online]. Avaliable: https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html .

[20] Logistic Regression, section:(1.1.11), Accessed: Aug.04, 2020.[Online]. Avaliable: https://scikit-learn.org/stable/modules/linear_model.html#id26 .

[21] M. Schmidt, N. Le Roux and F. Bach. Minimizing Finite Sums with the Stochastic AverageGradient. Mathematical Programming B, Springer, 2017, 162 (1-2), pp.83-112. hal-00860051v2.

[22] Random Forests, section:(1.11.2.1), Accessed: Aug.04, 2020.[Online]. Avaliable:https://scikit-learn.org/stable/modules/ensemble.html#forest .

[23] Decision Tree, section:1.10, Accessed: Aug.04, 2020.[Online]. Avaliable: https://scikit-learn.org/stable/modules/tree.html#tree .

[24] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification.

[25] Accuracy score, sction:(3.3.2.2), Accessed: Aug.07, 2020.[Online]. Avaliable: https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score .

[26] Log loss section:(3.3.2.11), Accessed: Aug.07, 2020.[Online]. Avaliable: https://scikit-learn.org/stable/modules/model_evaluation.html#log-loss .

[27] Adam Optimizer, Accessed: Aug.07, 2020.[Online]. Avaliable: https://keras.io/api/optimizers/adam/ .

[28] V. Lokare, S. Birari,M. Dang, and Prasad Bhagwat. (2016). Smart Virtual Assistant a Broad Perspective.

APPENDICES

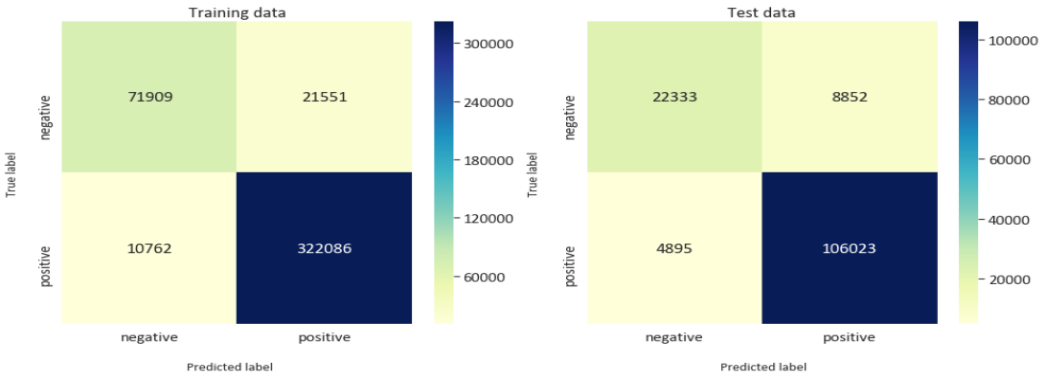

Figure 20: Confusion Matrix (Logistic-Bigram Model)

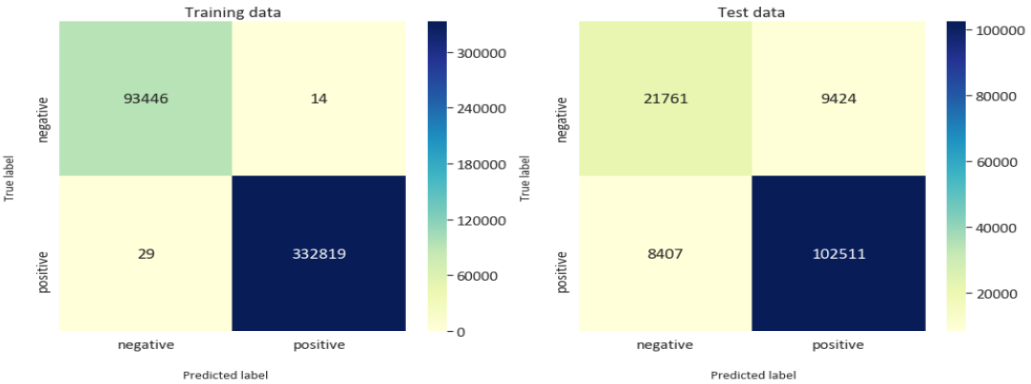

Figure 21: Confusion Matrix (Decision Tree)

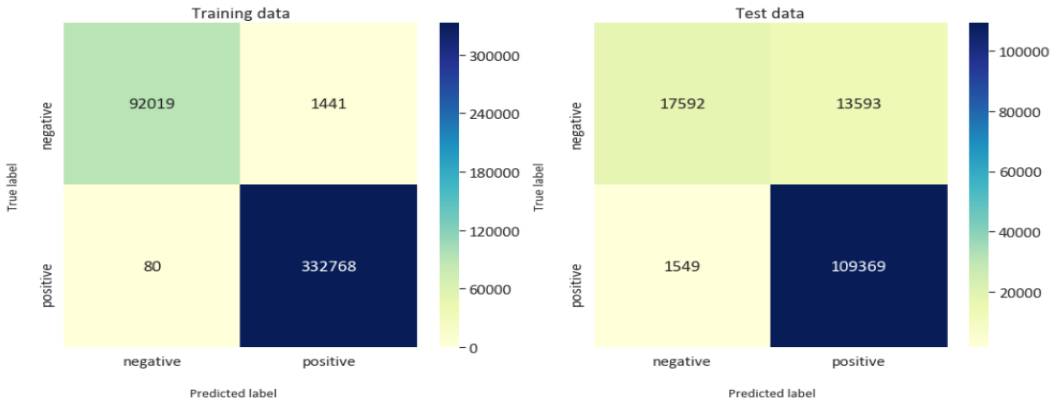

Figure 22: Confusion Matrix (Random Forest)

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	(None, 150)	0
embedding_2 (Embedding)	(None, 150, 200)	16018600
conv1d_9 (Conv1D)	(None, 150, 512)	102912
max_pooling1d_9 (MaxPooling1	(None, 149, 512)	0
conv1d_10 (Conv1D)	(None, 149, 256)	131328
max_pooling1d_10 (MaxPooling	(None, 148, 256)	0
conv1d_11 (Conv1D)	(None, 148, 256)	65792
max_pooling1d_11 (MaxPooling	(None, 147, 256)	0
conv1d_12 (Conv1D)	(None, 147, 128)	32896
max_pooling1d_12 (MaxPooling	(None, 146, 128)	0
flatten_3 (Flatten)	(None, 18688)	0
dense_5 (Dense)	(None, 128)	2392192
dense_6 (Dense)	(None, 2)	258

Total params: 18,743,978
Trainable params: 18,743,978

Figure 23: CNN model (Considering One word)

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	(None, 150)	0
embedding_1 (Embedding)	(None, 150, 200)	16018600
conv1d_6 (Conv1D)	(None, 150, 512)	102912
max_pooling1d_6 (MaxPooling1	(None, 150, 512)	0
conv1d_7 (Conv1D)	(None, 150, 256)	131328
max_pooling1d_7 (MaxPooling1	(None, 150, 256)	0
conv1d_8 (Conv1D)	(None, 150, 256)	65792
max_pooling1d_8 (MaxPooling1	(None, 150, 256)	0
conv1d_9 (Conv1D)	(None, 150, 128)	32896
max_pooling1d_9 (MaxPooling1	(None, 150, 128)	0
flatten_2 (Flatten)	(None, 19200)	0
dense_3 (Dense)	(None, 128)	2457728
dense_4 (Dense)	(None, 2)	258

Total params: 18,809,514
Trainable params: 18,809,514

Figure 24:CNN model (Considering Two words)