Designing classification and topic extraction system for online user reviews and analysing effect of meta data on classification

Ashish Kannur

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Data Science)

Supervisor: Prof. Bahman Honari

September 2020

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Ashish Kannur

September 7, 2020

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Ashish Kannur

September 7, 2020

Acknowledgments

I would like to express my gratitude and appreciation to my supervisor, Prof. Bahman Honari, for his constant support and encouragement in making this dissertation a reality. It was truly an honor having worked under his guidance and am thankful for all the insightful discussion sessions.

Through this project I was able to learn about the concepts of NLP in detail and Prof. Bahman guided me step by step to conduct an effective research. He patiently answered all my queries, technical and non technical.

I would also want to thank all my professors who taught me throughout the year and my friends for their constant support.

Ashish Kannur

University of Dublin, Trinity College
September 2020

Designing classification and topic extraction system for online user reviews and analysing effect of meta data on classification

Ashish Kannur, Master of Science in Computer Science

University of Dublin, Trinity College, 2020

Supervisor: Prof. Bahman Honari

Online platforms have opened up ample of opportunities for businesses to gain knowledge about the quality of their products and customers feedback through the examination of online reviews of the users. This study implements a sentiment classification and topic extraction system which accurately classifies the Amazon fine food online reviews into binary categories, positive and negative. Sentiment classification is performed via 2 approaches: Vectorized features processed with legacy Machine learning algorithms and deep learning techniques. Various algorithms like logistic regression, Support vector machine classifiers, decision tree classifiers are compared by measuring their performance in terms of how accurately they classify the reviews into positive or negative by learning from a labelled dataset and it is found that logistic regression performs the best on combination of review text and summary text. In this experimental study, the vectorization methods offer the flexibility of choosing or not choosing the order of words through the n-grams concept. In the non- neural network approach, vectorization methods used are countvectorizer and tfidfvectorizer which convert the review texts into document term matrix(numerical form) which is then used as input to drive the algorithms of machine learning. These methods are combined with the

balanced and unbalanced datasets along with and without the extracted text meta data to examine the effects on the model accuracy in classifying the reviews. While in the neural network approach, we use the LSTM and GRU networks which take into account the word order or word semantics. Here, we test the role of pertained models like Glove embeddings in the classification of the reviews into positive or negative. Again, these tests are performed on balanced as well as original datasets along with and without the extracted text meta data. It can deduced from the results obtained, that deep learning approach using the LSTM performs the best on the labelled dataset and also that model performance is not much affected by the distribution of prelabelled data or the data skewness. It is also seen that combining the meta data such as review polarity, review length, review character length, etc with the reviews enhances the binary prediction accuracy. As second part of this study, the topic extraction system is implemented which categorizes the reviews into fixed numbers of categories where is each category is a collection of similar and related words which when combined, give an intuition about a topic that the category is speaking about. This task of interpretation is however manual.

Contents

List of Figures

Chapter 1

Introduction

Product marketing and user feedback are efficiently comprehended using online technologies including various social networking platform, which give an easy hands-on to the companies. The ever increasing usage of these online platforms has reformed the way people do their shopping. Users have better and faster accessibility to information about the products through online mediums. Any information about the product can be obtained first hand on the relevant websites and is shown to be trusted more as compared to the information provided by the vendors. Online reviews are found to be more oriented towards the users and seem to share the real time practical experiences with the products. The consumers thus have an opportunity to read through plethora of reviews, examine them and then make informed decision about products they want to buy.

Most of the buyers withhold a feeling about an item by pursuing only one to three reviews. Whether it is a film, vehicle or a café, individuals always wish to comprehend what others are stating. As per Reevoo details, customer reviews contribute up to 18% of the total sales. The truth is, brands, have a more to pick up from positive audits and significantly than to lose from negative audits. It is easy to manually review a less number of customer feedbacks, however constraints elevate exponentially with an increase in the number of reviews. Hence, inorder to circumvent this, Sentiment Analysis (SA) becomes more relevant, helping to find the hidden context in the content [1]. SA is a significant domain in Natural Language Processing (NLP) for extracting subjective information from a piece of text and deciding the polarity of the user opinions.

Data generation from user experiences for brand selection and consumption has boosted with the widespread usage of social media. SA has received considerable attention as it turns unstructured user feedback into valuable details [2]. Extracting useful information from huge amounts of data is the need of the hour and for this purpose machine learning techniques are employed.[3].

1.1 Motivation

Using SA, companies can accurately analyse the sentiments of the reviews to derive insights about the product performance in the market. SA has a wide variety of applications like determining customer segments having the strongest opinions about a product, planning product improvements according to the received customer feedbacks, prioritizing customer service related issues and resolving them, thus preventing the spread of negative product reviews. Further, SA also aids in tracking customer sentiments about the products over a period of time, enabling prediction of the product lifecycle and also identifying the most effective communication media channel. Owing to the advantages that SA offers, its becomes relevant to incorporate it in the business models, for short and long term company benefits.

1.2 Problem Statement

Consumers who have actually bought the products give ratings to the product based on their experience in the range of 1 to 5, outlining the feelings of the users for a product.But, sometimes ratings and comments of the users do not justify each other. This research study aims at developing a system to accurately determine the polarity of online user reviews as positive, negative or neutral by scanning plethora of customer feedback on the products. We also analyse the factors such as meta data of the review texts that might affect the classification accuracy. Last but not the least, it is also important to know what exactly the users are talking about. A system is thus implemented to divide all the reviews into limited categories. The algorithms that are used for text grouping are Logistic Regression (LR), Support Vector Machine (SVM) and

Random Forest Classifier(RFC). In this study, preprocessing of text is also focussed upon because it helps avoiding the overfitting issue in the models and decreases the computational complexity by removing unnecessary content from the text.

1.3 Research Objectives/Questions

Through this study, we answer following research questions:

1. How accurately can we classify the reviews into positive, negative and neutral using best machine learning practices, deep learning approaches to see which performs better?

2. Are the text reviews enough to accurately classify them or extracted meta data add to the accuracy level of prediction

3. Do the derived sentiment polarity and ratings justify each other?

4. What are the users exactly talking about or can we categorize the reviews of the users into finite topics?

5. Studying the effect of balanced/unbalanced data on prediction.

1.4 Thesis Organisation

Rest of the thesis is arranged as follows: Chapter 2 is about the works done in the past and gives a brief overview of techniques applied for similar research studies. Chapter 3 gives a brief overview of the dataset used. Chapter 4 is about strategic methodology and approach adopted to implement this system along with the methods of model training and validation. Chapter 5 describes the step by step investigation of the results obtained. Final chapter is about the conclusions, limitations and future scope of this research study.

Chapter 2

Literature survey and Background

Text mining and extracting meaningful content from the available data is the most explored area in data handling domain. Prior to internet there was very less access to peoples' opinion about the products and hearing about the feedback of the users was possible only through personal contact. If an organization or company needed to find out the opinions of their products in the market then public surveys was their only option.After the introduction of Web, there has been an explosion in the user produced content which has boosted the techniques for opinion gaining [4]. A portion of the machine learning methods for SA like Naïve Bayes, Support Vector Machines has been examined in the paper [5].

Opinion mining from reviews is an unpredictable cycle, which requires something beyond text mining procedures. The unpredictability is identified with several issues. To start with, information of reviews are to be pooled from sites, in which web bugs or web crawlers can assume a significant job. And it is necessary to isolate the reviews from non reviews and then classification be performed based on the sentiment [6]. According to Pang et al.[7], algorithms of text mining perform well in traditional topic based categorization but are not effective in sentiment classification.

In a survey conducted by Crawford et al [8] on spam detection, different text features extracted through NLP techniques are discussed. Reviews are classified into spam and genuine using various perspectives. Classification was performed using algorithms of

Naïve Bayes (NB), SVM, LR and Bag of words, POS tagging and term frequency. In all the algorithms, SVM seemed to perform well most of the times however LR was a strong contender for classification as well according to them. It was seen that combination of multiple features enhanced the performance.

Shojaee et al(2013) [9], in their study about deceptive opinions found that the spammers usually change the composing style and utilize less number of syllables in the words. Lexical,syntactic features were extracted which included average length of the sentence and token, number of capitalized letters and tokens. The algorithms used were NB and SVM. F-score was used to compare the performance of the features as well as to combine them. Even in this study, combination of features showed improvement in the accuracy of the models. Also, SVM performed better than Naïve Bayes for the reviews.

Some studies have proved that combination of word frequency and semantic features is more effective than either of them being used alone. Lilleberg et al. (2015) [10] in their study combined the TFIDF vectors and Word2vec and also various other combinations such as including the stop words. They assumed that TFIDF does not establish semantic relationship between the word tokens whereas Word2vec does. '20 newsgroup' dataset was used for this study and the results showed that combination of Word2vec and TFIDF output vectors excluding the stop words gave the highest accuracy. Using only the TFIDF output also proved to be effective in some cases as compared to the combination of Word2vec and TFIDF output.

When it comes to text classification, classical machine learning algorithms such as NB,LR and SVM perform better, according to Vinodhini et al. (2012) [11] . Joachims(1998) [12] in his study justifies why SVMs are the best choice for text sentiment classification. According to him, the way of functioning of SVM does not depend on the problem dimensionality and has the capability to handle huge features and hence is suited for text classification problems. These problems are generally linearly separable and so, SVMs are best to find such separators.

For the precise sentiment classification, numerous researchers have put forth attempts to combine the ML and DL based techniques. Zhang et al.(2015) [13] implemented

Convolutional neural network (CNN) along with pooling layers which is effective in mapping the variable length sentences to fixed vectors. Islam J. et al. (2016) [14] used CNN for predicting sentiments of text. Pre-trained model of GoogleNet is used here along with parameter tuning of biases and weights. Performance of CNN increases with its size and depth and thus a 22 layered model is used for this study. Use of Stochastic Gradient Descent and back propagation optimizes the model further. Here, the dataset of twitter is selected. Images were labelled using the MTurk (Amazon Mechanical Turk). Proposed system in this paper outperformed most of the existing methods without performance tuning. Also GoogleNet provided 9% increase in performance as compared to AlexNet which was used in the previous works. Further, a reliable and stable state was achieved using hyper parameter tuning.

A. Severyn et al.(2015) [15] proposed in their study a system of CNN for sentiment analysis of tweets where the focus is to initialize parameters of the CCN network. Word embedding is initialized and unsupervised tweets dataset is trained. In this model, activation functions, pooling, convolutional layers and softmax is used. Gradients are calculated using SGD. Dropout enhances the neural network through regularization. Advantage of CNN is that it eliminates the need of extracting the features explicitly. On the other hand it learnS this internally from data.

X. Ouyang et al.(2015) [16] proposed a seven layer model for sentiment classification. The framework developed by them use CNN and Word2vec for SA and converting text to vectors. Dropout, Normalization and PReLU are used to increase model performance. Dataset used was movie reviews text obtained from rottentomatoes.com and was labelled into 5 categories. This model out performed previous models based on Matrix Vector RNN.

RNNs have also seemed to have performed very well in terms of capturing sematic relationships between the text tokens and classifying the texts. C. Li et al in 2014 [17] implemented a RNDM to predict the labels positive or negative at sentence level. This model achieved high performance than SVM and NB. Dataset was about the movie reviews from Chinese website.

In the study conducted by H. Chen and W. Li(2014) [18] , they implemented a framework which detected the malware sellers. It's a deep learning based model developed for text classification to check the product quality based on customer feedback. The forum reviews was used as dataset. Training was done using Recursive neural tensor network (RNTN) and evaluation was performed by comparing with NB, SVM and KNN models. The implemented model in this study successfully predicted the malicious sellers and showed that carding sellers have less ratings as compared to malware sellers.

Now moving on to the topic modelling section, methodologies have been proposed for feature recognition from text data. These approaches are based upon supervised, unsupervised and double propagation. These methodologies have the constraint that they do not collect similar semantic expressions. Supervised methods demand lot of manual work for manually labelling the data and hence not efficient. On the other hand, unsupervised methods have proved to be much more effective in identifying similar feature words. Topic models based on probability have a variety of algorithms to extract the hidden features form a huge document collection. Such models assume that reviews are made up of different topics and every topic is made up of a distribution of words. These topic models are convenient in terms of organising and summarising huge datasets. When such models are implemented on document, a low dimensional polynomial distribution is generated and each such distribution is called topic. Topics then capture information for each word so that semantic structure can be prepared. One of the popular algorithms used now a days is the Latent Dirichlets Allocation (LDA). The approaches which are existing, are good enough to catch word semantics but they still consider the bag of words approach and focus on extraction of unigrams for preparation of topics [19]. Thus these models only extract unigrams for preparing topics. Another limitation is that the word order is ignored. There have been many studies to overcome these limitations. Wallach [20] used Bayesian hierarchical model and LDA and developed a framework which generates bigrams. This model only generates bigrams and ignores unigrams whereas the model proposed in my work takes into consideration unigrams, bigrams and ngrams. Steyvers and Griffiths [21] developed a co-allocation model. This model uses new parameters to generate unigrams or bigrams but still could not produce reasonable topics. The model takes into account

the sentence structure, co-occurrences of words and relations between them. Wang et al. [22] developed the TNG(Topical n grams) which decided whether to create ngrams based on the nearby words and co-occurrences and dealt with sentences rather than documents. It also assigned same topics to consecutive words for a sentence. It was an improvement over existing LDA co-allocation model.

Gruber et al. [23] assumed in their study that there is Markov relation between the hidden features. The Markov model was proposed by them which assumed that same sentence words belong to same topic and most probably subsequent sentences would belong to same topic. Although this model extracted topics by considering the word order, it did not work well for ngrams. In this paper, topic model is implemented by considering the phrases and unigrams which is realistic and useful for applications. It is assumed here that word features of a sentence form a hidden Markov chain. It is a new topic modelling unsupervised approach which extracts features by considering the structure of sentences and capturing multiword features.

Tu et al. proposed an ONMF technique. Unlike the current online topic recognizing strategies, the created topics in ONMF were composed in a various leveled structure. Additionally, the technique can follow the developing cycle of the topic progressive system, which was cultivated by adaptively including rising topics and eliminating blurring topics. Klein et al. [13] applied an exceptional way to utilize NMF to make a topic model about authors' contribution on forum on Reddit.com. They concluded that inside the forum there are sub populations of people based on their views on various topics and different experiences and background.

2.1 Vectorized Features

For predictive modelling, text must be first converted into a list of words(called tokenization). These words then need to be converted into integers or floating point numbers to be used as inputs to machine learning algorithms. This arrangement of features utilizes the vectorization cycle to change texts to vectors. There are 2 ways to do this.

2.1.1 Count Vectorization

Scikit-learn, a famous ML package provides Countvectorizer library to convert the given text data into a feature vector representing count of tokens. This a highly flexible feature for representation of text.

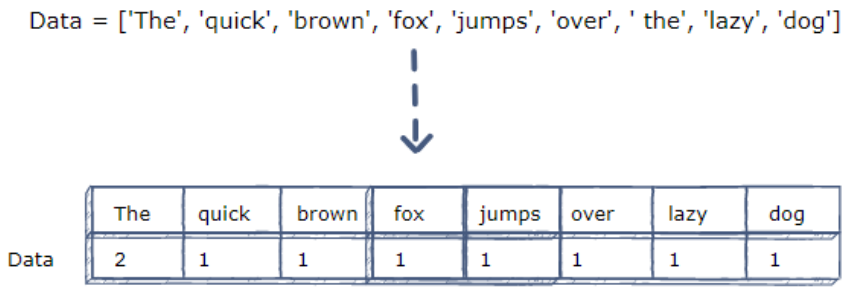

Figure 2.1: Formation of Count Vectorizer matrix elements for a single document [24]

	T1	T2	T3	T4
D1	F11	F12	F13	F14
D2	F21	F22	F23	F24

Figure 2.2: Count Vectorizer matrix for 2 documents

In Figure 2.1, preview of the working of count vectorizer is shown where the rows of the matrix are the documents or individual reviews while the columns are representing distinct words. Value of each cell indicates the number of times or frequency that a word has showed up in a specific record. For example, F11 means the recurrence of Term T1 shows up in Document D1. Matrix is prepared before giving it as input to the ML algorithm. As this technique records the occurrence of each word separately, it is not able to capture the relation between the words. This is where concept of n-gram comes into picture which gives the frequency matrix of group of 'n' words occurring together in a corpus. Example: bi-gram refers to frequency matrix of unique pair of

words occurring together in the specific reviews.

2.1.2 TF-IDF Vectorization

This vectorization method works similarly to countvectorizer method except that instead of frequency of the word it calculates the TF-IDF of the term in consideration. The numerical formulae's for evaluation of TF-IDF are given in figure 2.3.

$$TF(t,d) = f(t,d)$$

$$IDF(t,d) = \log N/(\ |d \in D: t \in T|)$$

Figure 2.3: TF-IDF Vectorization Formula

The basic idea behind TF-IDF term is to reduce the weights of the unnecessary terms which occur frequently and do not contribute much to meaning of the text. Here TF refers to term frequency whereas the IDF refers to inverse document frequency which is nothing but log of the ratio of total number of reviews/documents and number of documents in which the term occurs. Thus, more the frequency of the term , more is the denominator of the IDF ratio and hence less is the value of IDF, which satisfies the condition of giving less weights to the frequently occurring terms. 1 is added in the denominator to avoid divisions by zero for an unseen word and thus acts as a smoothing parameter. n-grams concept can be extended here as well.
As an example, consider a document containing 100 words wherein the word cat appears multiple times. The term recurrence (i.e., tf) for cat is at that point (3/100) = 0.03. Presently, expect we have 10 million documents and the word cat appears in 1,000 of these. At that point, the backwards document recurrence (i.e., idf) is determined as log(10,000,000/1,000) = 4. Hence, the Tf-idf weight is the result of these amounts: 0.03 * 4 = 0.12.

2.2 Review Centric Features

Features of text can be categorized into basic, semantic and syntactic features. Review word count/length and character count are basic features while features such as number of punctuations, capital words etc are syntactic features. The polarity and subjectivity of the review text fall into the category of semantic features.

2.3 Summary Text Features

Alongside the review text, summary of the review is also taken into account. Similar to the review text features, basic /syntactic and semantic features of the summary text can be extracted and used for this study.

2.4 Meta Data

Features apart from the semantic features are considered meta data. And in this dataset the meta data that is given to us is the rating or score the users have given to the products which is in the range of 1 to 5.

2.5 Classification Machine learning algorithms

2.5.1 Naïve Bayes

Naive Bayes assumes that features are always interdependent and represents the text in the form of multinomial distribution by converting it first to Bag of words. Naive Bayes belongs to the class of probabilistic classifiers (Aggarwal et al., 2012) [25]. Likelihood of each word is multiplied to obtain the likelihood of a document given a class. Then using the Bayes rule, posterior likelihood of the class when document is given is calculated and the highest value of the posterior likelihood of the class is assigned to the document.

For m unique classes and n-dimensional document vector, posterior probability is given as:

$$P(C_i \mid D) \propto P(C_i) \cdot \prod P(x_t \mid C_i) \qquad (i = 1,2,3, \ldots, m \text{ and } t=1,2,3, \ldots, n)$$

2.5.2 Logistic Regression

Logistic regression belongs to the class of linear classifiers and variables that have discrete responses are modelled using linear regression. We get the co oefficients of the regression model(A and b) by optimizing the function for conditional likelihood (Aggarwal et al., 2012) [25]. Posterior likelihood is calculated by the formula given below and highest likelihood value of the class is assigned back to the document as in the case of MNB model:

$$P(C_i \mid X_j) = e^{\wedge} (A \cdot X_i + b) \,/\, (1 + e^{\wedge} (A \cdot X_i + b))$$

By combining the features linearly and using the logistic function, the posterior probability can be transformed to linear regression problem.

2.5.3 Support Vector Machine

Support Vector Machine aims to isolate various classes by finding a separator in the feature space. Thus separator is called hyperplane. SVM uses optimization technique called quadratic programming to find the best hyperplane with margin of separation (Aggarwal et al.)[25]. Fig. 2.4 shows SVM algorithm in action.

2.5.4 Random Forest Classifier

Random forest classifier is formed by combining individual decision trees. Thus decision trees are the basic building blocks of Random Forest Classifiers. Decision trees work on the principle of splitting into branches based on certain condition in order to separate classes. An example of decision tree is given in the Fig. 2.5. The nodes which are at the bottom of the tree where the decisions are made are called leaf nodes[26] . As shown

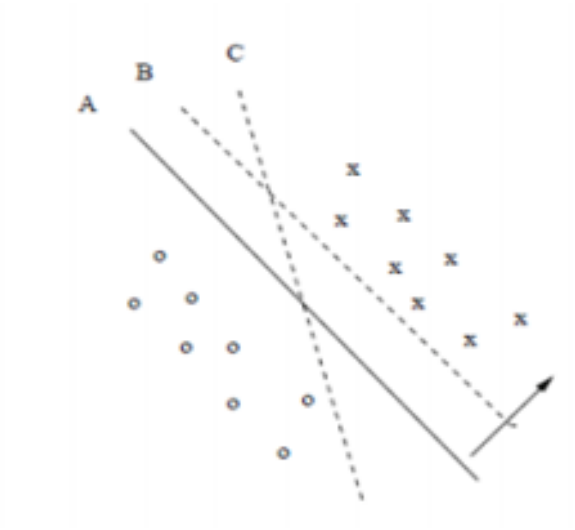

Figure 2.4: Hyperplane for SVM with separation margin [25]

Figure 2.5: Example of Simple Decision Tree [26]

in Fig 2.6,in the random forest, each tree makes a prediction of a class separately. The class which gets more number of votes becomes the prediction of the model. These models are uncorrelated and collectively these models outperform any individual model. For Random Forests to perform better, there must be low correlations among the trees and there must be some signal in dataset features so that models do not just do random guessing.

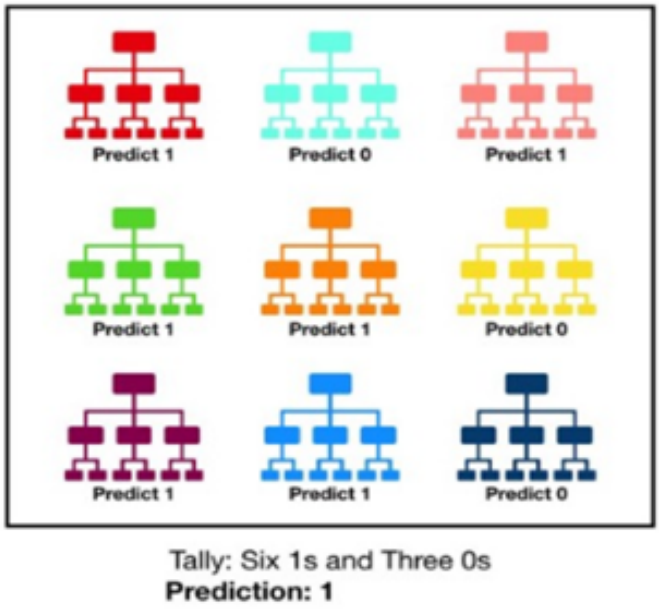

Figure 2.6: Visualisation of Random forest making prediction [26]

2.5.5 Glove Embedding

Glove is used to represent words in vector form by using unsupervised learning algorithm. Model is trained on collected worldwide word-word co-occurrences from a corpus and representations resulting from this show vector space's linear substructures. Semantic similarity between words is measured using the Euclidean distance between the word vectors which is very effective. This metric reveals rarely used but relevant nearest neighbours which sometime are out of normal human vocabulary. The relatedness between the two words is represented by a scalar which is produced by the

metrics of similarity used for nearest neighbours evaluation. Now this singular scalar number is not sufficient to represent the relationship between two words because most of the times, two words share more intricate relationships than which is represented by just a single number. Thus it is necessary to assign more than single more number to a pair of words. Vector difference between the vectors of words is thus chosen to capture the meaning or semantic relation between the pair of words. The distance between the different pairs are same if the amount of contrast between the words in each pair is similar[27] . This can be seen in the visualizations shown below in Fig 2.7. The pre-trained Glove model is developed by training it on the global word to word

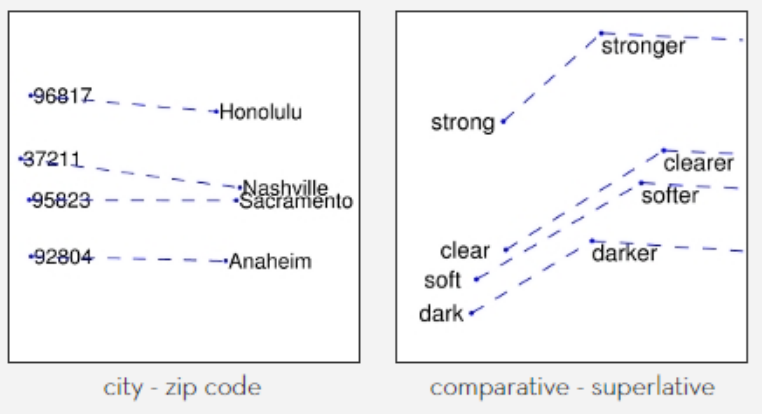

Figure 2.7: Representation of Glove Embedded word in feature Space [27]

co-occurrence matrix which is nothing but one representing the frequency of the word pairs occurring together. To prepare this matrix, it has to pass only once through the corpus which can be costly if the dataset is very large, however it is a one time computation. The intuition underlying the Glove model is that the ratio of co-occurrence probabilities represents significant meaning. The training of Glove is based on the fact that logarithm of the co-occurrence probability ratio is equal to the dot product of the word vectors. Only such word vectors are learnt by the model. Thus this model is most suitable for tasks related to word analogies.

2.5.6 LSTM(Long Short Term Memory)

Traditional neural networks lack in their ability to use the previous events to make predictions about the later ones. Recurrent neural networks overcome this issue by holding the information through loops. In fig 2.8, small chunk of the RNN network is shown along with its equivalent when the loop is unrolled [28] . The loop allows

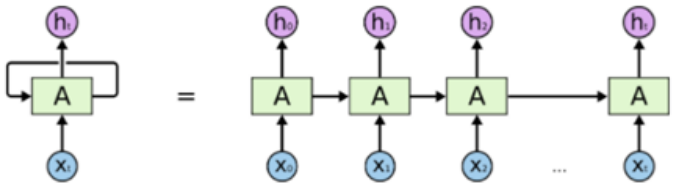

Figure 2.8: Unrolled RNN
[28]

to pass the information from one part of the network to another. As such, RNNs have found their success in many applications such as speech recognition, captioning of images etc. Now traditional RNNs are capable of using only the recent information to predict the next values. However when the value to predicted is dependent on a value which occurred way before the previous values, in such cases where the gaps are more, RNNs fail to perform well. LSTMs(a version of RNNs) solve this problem of long term dependency by remembering information for a longer period of time. In addition to the single tanh layer in the basic RNN, LSTMs have a combination of sigmoid and tanh gates as shown in fig 2.9. The line running throughout the top in

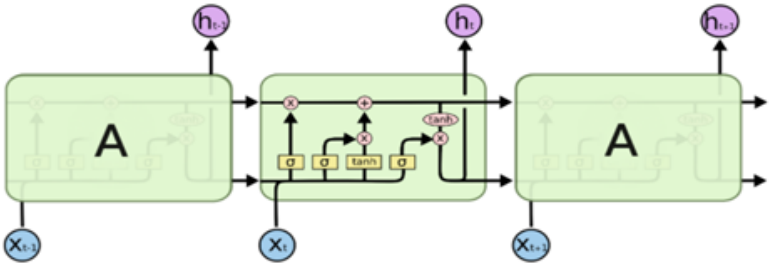

Figure 2.9: Interacting layers in LSTM module
[28]

the diagram tells us that the information flows unchanged through the entire chain

with little linear changes. Sigmoid gates control how much of the information should be let through. This also introduces the concept of forgetting the information. Based on the cell state of previous stage and current input, the network decides to forget the information or retain it.Thus, network only remembers the relevant information and forgets the irrelevant ones.

2.5.7 GRU(Gated Recurrent Unit)

GRU is constructed on the same logic as that of LSTM just that it has gotten rid of the cell state. Information from one cell to another is transferred using the hidden state. Reset and update gate is introduced in this model.

2.6 LDA(Latent Dirichlets Allocation)

Topic modelling refers to deriving hidden patterns and topics from the given text corpus. Topic modelling, a field of text mining, is different from dictionary based keywords searching techniques and is a unsupervised approach to identify the clusters of semantically related words in a huge corpus. It has wide range of applications like arranging large datasets of emails, social media or online product reviews. Latent Dirichlets Allocation is the most recognized topic modelling algorithm. It is assumed in this model that each document is mixture of topics. Words are generated by the topics using the probability distribution. LDA uses backtracking to recognize what topics would generate the document. Document term matrix, basically the output of TF-IDF vectorization, is used as input to LDA algorithm. LDA splits this matrix to matrices of lower dimensions as shown in fig 2.10 [29]. Here the number of documents is n, number of unique words or features is m and number of topics chosen is K.

Figure 2.10: Matrix formation in LDA algorithm
[29]

LDA keeps improving these matrices by iterating through each word w of each document d and attempting to alter the topics. Initial states are randomly chosen. Topic K is assigned to a word w based on a likelihood calculation P. P is a combination of two probabilities viz. probability of topic given a document and probability of a word given a topic. These 2 probabilities are multiplied to get likelihood P. Thus we get the likelihood of a particular topic producing a particular word. After many such iterations, a stable state is achieved and the distribution obtained is acceptable [29] .

2.7 NMF(Non Negative Matrix Factorization)

NMF is similar to LDA. The manner in which it works is that NMF decays (or factorizes) high-dimensional vectors into a lower-dimensional portrayals. These lower-dimensional vectors are non-negative which likewise implies their coefficients are non-negative. Utilizing the first grid (A), NMF will give both of you frameworks (W and H). W is the themes it found and H is the coefficients (loads) for those points. At the end of the day, An is articles by words (unique), H is articles by themes and W is subjects by words [30] . Matrix split and calculations are shown in fig 2.11 and 2.12.

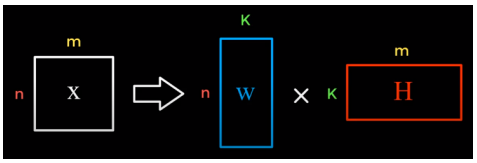

Figure 2.11: Matrix formation in NMF algorithm
[31]

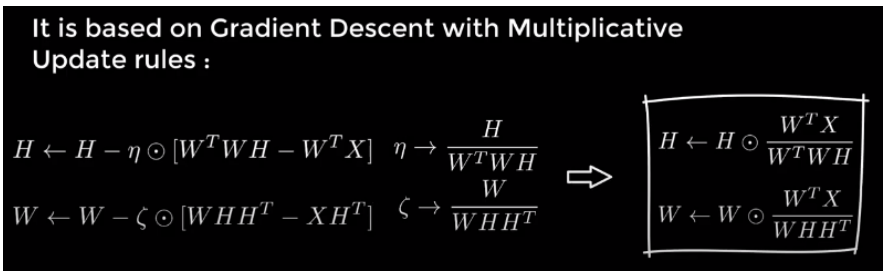

Figure 2.12: Matrix calculation in NMF
[31]

Chapter 3

Synopsis of the data

The dataset used for this study is about the Amazon fine food reviews posted on Amazon.com. This data was first distributed by McAuley et al. (2013) [32] while studying about online surveys. There are total 568,454 reviews on about 74,258 items. Detailed description of the dataset is given in the table below:

Review attribute	Description	Variable type
Product ID	Unique identifier for the product	Categorical
User ID	Unique identifier for the user	Categorical
Profile Name	Profile of the user	Text
Helpfulness Numerator	Number of users who found the review helpful	Numerical
Helpfulness Denominator	Number of users who voted whether the review was helpful or not	Numerical
Score	Rating between 1 and 5	Ordinal
Time	Timestamp of the review	Numerical
Summary	Brief summary of the review	Text
Text	Text of the review	Text

Figure 3.1: Data Overview

3.1 Review Text and Summary Text overview

The reviews contain the actual description of the item in detail from the customers' viewpoint. In addition, it likewise portrays the general sentiment or experience of the user with the item in comparison with the description. Some examples of reviews and summary texts given in the dataset:
Review Texts:

1. I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.
2. Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. Not sure if this was an error or if the vendor intended to represent the product as "Jumbo".
3. If you are looking for the secret ingredient in Robitussin I believe I have found it. I got this in addition to the Root Beer Extract I ordered (which was good) and made some cherry soda. The flavor is very medicinal.

Summary Texts:

1. "Delight" says it all
2. Great! Just as good as the expensive brands!
3. GREAT SWEET CANDY!

In this research study, one of the methodologies utilizes the features extricated from the summary and review texts so as to identify if the features of the text such as average word length, total character length, total number of words, sentiment polarity score (between -1 to 1) are helpful in identifying the sentiment of the review.

3.2 Score Positivity

Score is the rating given by the user to the product. The range of score is ranging between 1 to 5 where 1 would denote lowest rating and 5 would be highest rating. Generally scores 1 and 2 denote dissatisfaction of the user towards the product whereas scores 4 and 5 represent that user is satisfied with the product. Score of 3 shows neutral sentiment of the user for a product. Hence in our study scores of 3 is not considered as it does not contribute much to sentiment derivation from the reviews. That being said, the reviews are categorized into 3 main categories based on the scores. Reviews with Scores 1 and 2 would be labelled as 'negative' , Score 3 reviews would be labelled neutral and reviews with scores 4 and 5 would be labelled positive. Thus, instead of predicting the scores of each review, our classification task reduces to predicting reviews as positive, negative or neutral.

3.3 Text Semantics

In this approach of centric features of review and summary text, we attempt to find if there is any effect of the features extracted from text on predicting the sentiment o the review. One such feature used is the sentiment polarity. This feature is extracted using python API called TextBlob. If we peek into the source code of TextBlob package, we can see that it uses Naïve Bayes Classifier to categorize the reviews into positive and negative categories based on probabilities. Individual scores of words are referred to from the sentiwordnet and average polarity is calculated.

Chapter 4

Methodology

In this section, it is described in detail the strategy, approach and methodology to build automatic topic extraction and text classification system. The dataset used is of Amazon Fine Food reviews. Brief overview of the steps involved in this process are(fig 4.1):

• Performing exploratory information investigation for creating the double reaction variable

• Explaining different content pre-processing steps which are utilized to eliminate unwanted terms

• Describing the topic modelling system, classification system along with their methodologies.

• Description of model training and model validation techniques.

4.1 Strategy and Approach

This section describes methodological approach or the thought process to achieve the objectives:

• For feeding the text to the classical machine learning algorithms, Vectorization methods would do our job of converting the text into numerical form i.e. convert the text to collection of vectors called as document term matrix.

• The document term matrices obtained in the above step is then used by LDA or NMF algorithms for topic extraction.

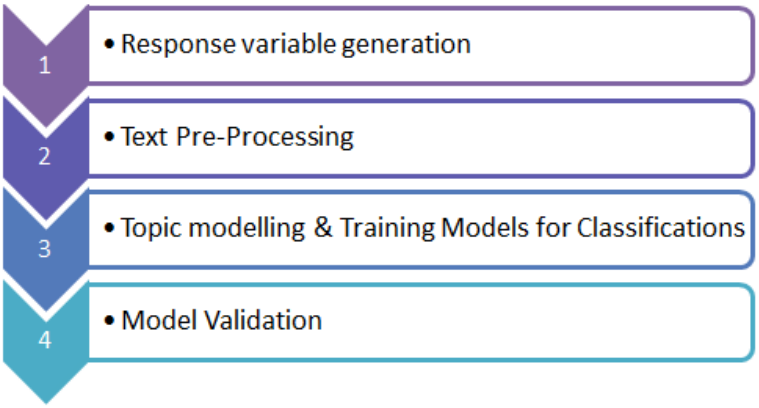

Figure 4.1: Workflow Diagram

• The output of the vectorization methods would be compatible with the other numerical metadata such as length of reviews, Character count, Sentiment polarity.
• For feeding the text to the neural network, it should be converted into an array of numbers. For this we can follow the dictionary based approach or the one hot encoding approach.

4.2 Text Pre-processing

Text pre processing is the most basic and very important step in designing text classification system. Text pre processing inshort refers to removing unwanted features of the text for efficient and faster computation and to eliminate the overfitting issue.The elements of text pre processing are: Tokenization, normalization and Substitution [33] .

4.2.1 Tokenization

Tokenization, known as text division, is process of breaking down of strings of sentences into separate parts called tokens. The division of sentences is mostly done into separate words. This may seem pretty staright forward but it is not. First question that we may think is how are sentences recognized in huge texts. All things consid-

ered you presumably state "sentence-finishing accentuation," and may even, only for a second, imagine that such an announcement is unambiguous. For example, you have a simple sentence ending with a full stop. That's pretty easy to segment. But what if the sentence has more than one full stops. It should be instinctive that there are differing procedures for distinguishing segment limits, yet in addition what to do when limits are reached. For instance, we may utilize a division technique which (effectively) distinguishes word tokens as punctuation. [33] .

4.2.2 Normalisation

Prior to additionally handling, text should be standardized. Standardization by and large means to bring all the content on a level. Standardization puts all words on equivalent balance [33] . Normalizing text can mean playing out various assignments, however for our structure we will move toward standardization in 3 steps: stemming,lemmatization and other cleaning steps .

4.2.3 Stemming

Stemming refers to taking out the suffixes, prefixes, etc from a word so as to get word stem.

4.2.4 Lemmatization

Lemmatization refers to catching authoritative structures of a word. It is somewhat similar to stemming only that stemming neglects the reference structure of the word whereas lemmatization reduces the word to its similar form or its base version called lemma. For example, lemmatization of word 'better' would be 'good'. We use python library Wordnetlemmatizer for this task [33] .

4.2.5 Other cleaning steps

Stemming and lemmatization are significant pieces of a book preprocessing. These aren't straightforward content control; they depend on definite and nuanced comprehension of linguistic guidelines and standards. There are many ways to bring a balance in the content and basic ones would be exclusion and replacement. They are, be that as it may, no less critical to the general cycle. These include:
• converting characters to their lowercase form.
• conversion of numbers to words or simply eliminating them
• eliminate accentuation (by and large piece of tokenization, yet worth remembering at this stage, even as affirmation)
• stripping void area
• removing the stop words such as 'the', 'and' , 'but' etc which occur most frequently. Stop words do not hold much meaning.
• eliminate given (task-explicit) stop words
• eliminate inadequate terms (not generally important or supportive)
Text preprocessing depends vigorously on pre-assembled word references, information bases, and rules [33] .

4.3 Noise Expulsion

The replacement tasks of the preprocessing framework is noise removal. While the initial 2 significant strides of our framework (tokenization and standardization) were commonly relevant as-is to about any content piece , noise expulsion is more task specific segment of the framework. General steps in noise removal are:
• eliminate text record headers, footers
• eliminate HTML, XML, and so on markup and metadata
• extricate important information from different configurations, for example, JSON, or from inside data sets
• In this part of pre processing most use of regular expressions is done.
Thus noise expulsion must be performed before handling of other steps of text pre processing. For example, when the data is acquired in JsOn format, it is in raw format

and has many unnecessary components like HTML tags, etc which should be taken out before tokenization [33] . Following figure 4.2 depicts the data cleaning process which reduces the whole review to its basic form.

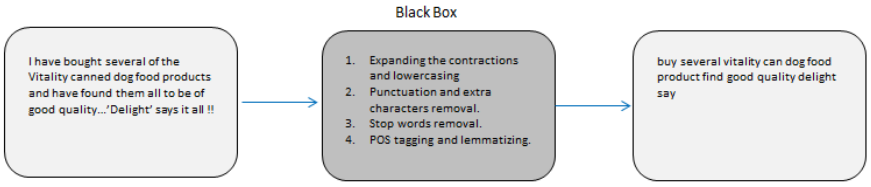

Figure 4.2: Text Cleaning Process

4.4 Exploratory Data Analysis

In this section, some pre manipulations of dataset before model training are discussed. The above figure 4.3 depict the distribution of score or the ratings given by the users.

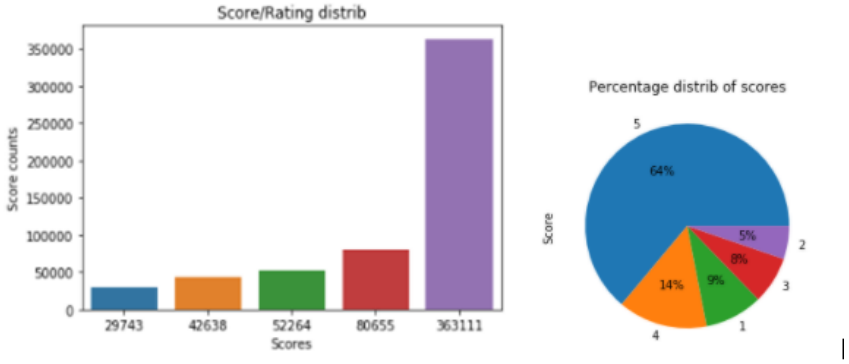

Figure 4.3: Score Distribution in given dataset

We can see that the number of reviews with score of 5 are more than reviews with scores 1 to 4 or in other words the data is highly skewed towards the score 5. In this study, we ignore the reviews with scores 3 as they express neutral sentiments and do not contribute much to the sentiment analysis of such reviews. After ignoring the neutral reviews, we have the distributions as shown in below fig 4.4(left). Fig 4.4(right) shows the count of the positive and negative reviews after they are made equal (undersampling) for removing the possibility of bias from our prediction model. In the

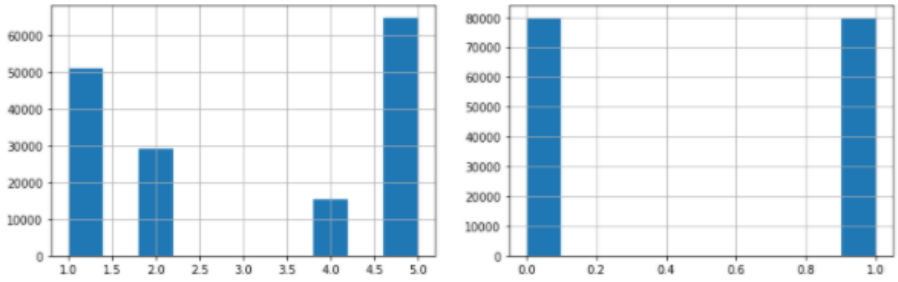

Figure 4.4: Score counts to the left after ignoring the neutral reviews and Count distribution after undersampling of positive and negative labels

below figure 4.5, the average length of the reviews for grouped by scores is shown. Average length of reviews ranges from 230 to 300. Next we go on to analyse the top

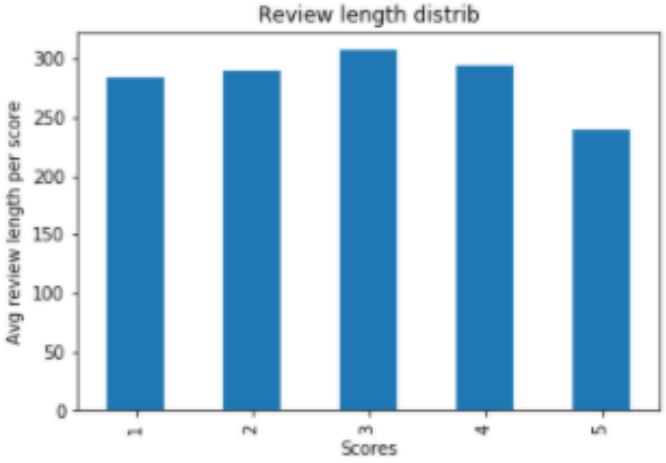

Figure 4.5: Review Length Distribution

words occurring in the reviews for each scores. Following figure shows the top 25 words occurring in the reviews with score 5. As the score is 5, it must represent a collection of all positive words and same is seen in the figure 4.6. Words like, love, taste, good etc occur more number of times. Similarly, if top words for score 1 or 2 reviews is plotted then we would see the negative words occurring most number of times. The output of countvectorizer is most suitable for this type of analysis as it is a frequency or term matrix. Now if we want to do the same analysis as above for every product then the input to the above logic would be the reviews of particular product and visualization obtained is shown in figure 4.7. In the given dataset, product Ids are given instead

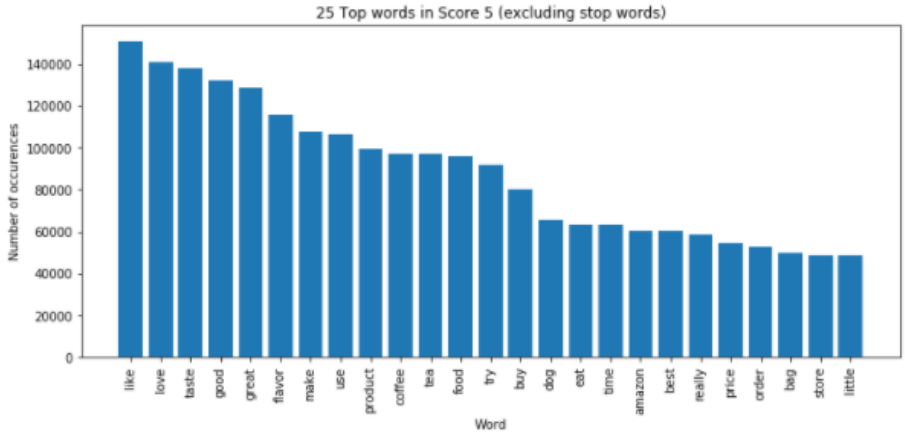

Figure 4.6: Top Words in Score 5

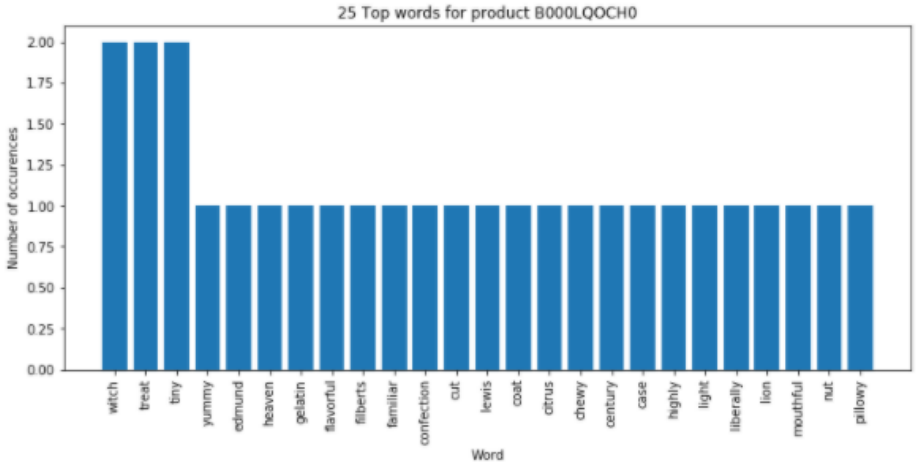

Figure 4.7: Top words for a example product

product names. The above figure is for unigrams. The same analysis can be performed for bi grams or n-grams for getting better understanding about what users think about the product.

4.5 Model Training

In this research study, we use the combination of Review text/ Summary text/ Meta data (given and extracted from the text) along with two vectorizers and 3 balancing methods (SMOTE / Undersampler / Oversampler) and 3 famous machine learning algorithms(LR-Logisitic Regression/SVM-Support Vector Machine/RFC-Random Forest Classifier). This approach creates around 40 scenarios. The basis for comparing these scenarios is accuracy. In the first approach, vectorization is performed on the text and converted to matrix form with rows of the matrix representing the reviews while columns representing the distinct words in the corpus.If the vectorizer used is count type, then the terms of the matrix would be term frequency and if the TF-IDF vectorizer is used then terms of the matrix would be TF-IDF terms as described previously in chapter 2. Additionally n-grams(unigrams , bigrams, trigrams) are extracted using these vectorizers for establishing the semantic relationship among the words. Figure 4.9 gives a summary of all combinations in this approach. In the secondary part of the

Feature	Description
Countvectorizer with unigram	Word frequency matrix for singular unique words
Countvectorizer with unigram +bigram	Word frequency matrix for pair of unique words
TF-IDFvectorizer with unigram	Combination of inverse document frequency and word frequency for singular unique words
TF-IDFvectorizer with unigram + bigram	Combination of inverse document frequency and word frequency for unique words and pair of words

Figure 4.8: Summary of Vectorizers combinations

approach, the feature set used consists of features related to structural, syntactic and sentimental and the meta-data. Features are extracted from Review text and summary

text both with a goal to find the suitable set of features which can contribute to the accuracy of review sentiment polarity prediction. Text pre processing should not be performed for extracting features like the structural and syntactic. Figure 4.10 gives a summary of the features. For deriving the polarity of the reviews, Textblob python

Feature	Example of the feature
Structural	1. Words count of the review/summary text 2. Character count of the review/summary text 3. Average word length of the review/summary text
Syntactic	1. Count of punctuations in the review/summary text 2. Count of uppercase words in review/summary text
Sentimental	1. Polarity of the review/summary text
Metadata	Score or rating

Figure 4.9: Type of features

library is used whose interal working is described in Chapter 3 – Text Semantics.
Once the features are extracted and text pre processing is done, it is optional to perform normalization of the vector matrices for scaling. Normalized values are easy to deal with and make the model more optimized. For example,logistic regression uses gradient descent method to maximize the likelihood function. Scaling of these features is in the range [0,1] or [-1,1].
For the neural network based approach, input to the neural network should be in the form of arrays of numbers and so we apply dictionary based approach or one hot encoding approach. In the dictionary approach, the unique words in the corpus are arranged in a dictionary. The pre processed reviews are then represented by the index numbers of the words from the dictionary. Thus the reviews are converted into a representation of series of numbers which is then converted into an array form and given as input to the neural networks. Also the array size of all the reviews is made same using padding zeros as it is the requirement of the neural networks for the input dimensions to be of same size.
In another approach for neural networks, one hot encoding is performed on the dataset reviews and converted to matrix with rows representing the documents/reviews whereas

columns/features are the distinct words occurring in the entire corpus. Even this matrix has proved to be more useful for training model using neural networks.
The two neural networks algorithms were use in our study are the GRU(Gated Recurrent Units) and LSTM(Long Short Term Memory). These have been described in detail in the previous Chapter 2.
For topic modelling, the input to the algorithm is the TF-IDF output or the countvectorizer output. LDA and NMF model working is described in detail in Chapter 2. Scikit learn provides python package for LDA and NMF. The number of topics needed can be given as parameter to these algorithms. Parameter tuning can be done by adjusting the values of alpha and beta.

4.6 Validation of models

This section discusses the validation of models and measurement of the performances so as to compare the superiority of the models.
This study splits the dataset into 7:3 ratio i.e. training and testing datasets respectively. Models are first trained and prepared using the training dataset and then validate by running the model through the test dataset. This technique is known as hold out techique. In some models, k-fold cross validation is also performed to overcome the overfitting issue. The sentiments of the reviews are predicted into binary classes. Henceforth, the possible presentation measures for this examination incorporate specificity, sensitivity, recall, precision and region under ROC curve. In this binary classification task we perform the evaluation of performance through confusion matrix. The confusion matrix parameters can be described as follows:

	Predicted positive	**Predicted negative**
Actual positive	True positive	False negative
Actual negative	False positive	True negative

Figure 4.10: Confusion Matrix

• TP– test case belonging to positive class anticipated as positive

• TN – test case belonging to negative class anticipated as negative
• FP – test case belonging to negative class anticipated as positive
• FN - test case belonging to positive class anticipated as negative

Evaluations of different performance measures of this binary classification can be explained by the parameters. In below Fig 4.11 we explain these parameters. The objective of this study is to classify the reviews into binary categories and study the effect of meta data on the classification accuracy. Thus, model performance validation is the necessity. As it very well may be deduced from Fig 4.11, the accuracy is characterized

Measure	Formula	Evaluation
Accuracy	$(tp+tn)/(tp+tn+fp+fn)$	Effectiveness of accuracy
Precision	$tp/(tp+fp)$	Agreed positivity measure achieved
Recall(Sensitivity)	$tp/(tp+fn)$	How effectively can classifier identify positive labels
F-Score	$(\beta\,2+1)\cdot tp/[(\beta 2+1)\cdot tp+\beta 2\cdot fn+fp]$	Relation between classifier's positive labels and positive labels in the dataset
Specificity	$tn/(fp+tn)$	Effectiveness of classifier to identify negative labels
AUC	$1/2[(tp/(tp+fn))+(tn/(tn+fp))]$	Ability to avoid wrong classification

Figure 4.11: Parameters of Confusion Matrix

as the proportion of accurately grouped examples to the total test cases.It represents classifier's overall performance.

Performance measure for the topic models is perplexity and coherence. Perplexity of a model is the log likelihood which is normalized. It is basically the measure of how surprised the model would be when it sees new data i.e. the probability of unseen data based on the previously learnt model. Coherence is measured by the semantic similarity among the words which are highly scoring. Words or sentences are said to be coherent if they support each other. Most commonly used coherence measure is the C_v which is a sliding window based one set segmentation of top words which uses NPMI(normalised point wise mutual information) and cosine similarity. The hyper parameters for tuning are the number of topics, document topic density(alpha) and word topic density(beta). These hyperparameters are varied in combinations and the optimal parameters [34] .

Chapter 5

Analysis and Evaluation

This area examines the examination performed utilizing the features which are vectorized. In this study, featured matrix is formed by converting the review texts to document matrix form where the rows represent review records and the columns are the unique words in the whole corpus with count vectorizer and TFIDF vectorizer as two simplified lattice variations. In check vectorizer, grid component indicates frequent bag of words occurance that shows up in a specific corpus, though component of a TFIDF grid contains values got by product of the term recurrence with iIDF value of a word for a particular review. Additionally, these matrices have in their columns singular words(unigrams) or pair of words(bigrams). Picking word pairs helps in revealing the hidden semantic connections among the available words. As an execution analysis method, both singular and multiple words are broken down with an initial text pre-processing to filter out the unwanted words, thus reducing the matrix size. Following this, the original data is randomly split into 70 percent training data and 30 percent test subsets using 'sci-unit learn' toolkit. Further, this library is also used for hyperparameter tuning.

For Logistic regression and SVM regularization parameter and L-1 penalty norm is used. Now let's discuss the results of different models trained using combination of review text/summary text/textual extracted meta data along with 2 vectorization methods, 3 balancing methods, 3 classifiers and finally with cross validation/without validation. Before we discuss the above table(Fig 5.1) results, lets have brief overview of the dataset balancing methods. Data skewness poses a grave problem in model

Input type	Vectorization	Balancing method	Method	Validation	Accuracy
Review text	Tfidfvectorizer(ngram)	Balancing using SMOTE	logistic regression	cross validation	86%
Summary	Tfidfvectorizer(ngram)	Balancing using SMOTE	logistic regression	cross validation	82%
Review text + Summary	Tfidfvectorizer(ngram)	Balancing using SMOTE	logistic regression	cross validation	88%
Review text + Summary + Meta data	Tfidfvectorizer(ngram)	Balancing using SMOTE	logistic regression	cross validation	65%
Review text	Tfidfvectorizer(ngram)	Balancing using SMOTE	SVM	cross validation	87%
Summary	Tfidfvectorizer(ngram)	Balancing using SMOTE	SVM	cross validation	88%
Review text + Summary	Tfidfvectorizer(ngram)	Balancing using SMOTE	SVM	cross validation	91%
Review text + Summary + Meta data	Tfidfvectorizer(ngram)	Balancing using SMOTE	SVM	cross validation	80%
Review text	Tfidfvectorizer(ngram)	Balancing using SMOTE	Random Forest Classifier	cross validation	84%
Summary	Tfidfvectorizer(ngram)	Balancing using SMOTE	Random Forest Classifier	cross validation	82%
Review text + Summary	Tfidfvectorizer(ngram)	Balancing using SMOTE	Random Forest Classifier	cross validation	85%
Review text + Summary + Meta data	Tfidfvectorizer(ngram)	Balancing using SMOTE	Random Forest Classifier	cross validation	83%

Figure 5.1: Table 1

training. Usually, the majority class of labels seems to dominate the prediction i.e. when a model is trained on highly skewed dataset and ran on a test data, then the model tends to predict most of the test dataset records with the label that is majority in the train dataset.

The first method for data balancing is the oversampling or up-sampling. This approach basically increments the size of the rare samples or increases the minority class members in the dataset. The main advantage of this approach is that no data is lost in this i.e. minority or the majority class. But by this approach, the model is prone to overfit. Another similar approach is Undersampling with the only difference being that this method lessens the majority class samples by popping out the samples from the original dataset. The advantage of this method is that it increases the memory efficiency and improves the run time. But this method causes loss of information.

SMOTE(Synthetic Minority Oversampling Technique) is another method used for handling data imbalancing. SMOTE comes into picture when oversampling of the minority class is done. This method developed by Nitesh et al(2002)[35],basically collects the samples that are in the vicinity of the current point in the feature space and the selects a point between current point and one of the points selected from a group of points in the vicinity. Such points created make more sense as they are close to the existing samples. There are couple of other variations in SMOTE for eg. Borderline SMOTE, SMOTE-SVM, ADASYN whose scope is beyond this research study. Now coming back to the results obtained as shown in the table (Fig 5.1), the vectorizer used to train and test all the models is TF-IDF vectorizer. Additionally, n-grams(singular words, pair of words, triplets) is considered to establish the semantic relationship among the words more effectively. As input to the model, only review text or concatenation of review and summary text along with or without extracted meta data like the number of characters in the particular review or average length of the words or number of words in the review or number of punctuations or sentiment score is given. These models are trained using each of the classifiers viz. LR, SVM and RFC are used. These models are evaluated by Accuracy measure which is nothing but the total number of positives and negatives predicted correctly out of the total number of reviews. Lets not forget that in all the cases, the data is balanced using the SMOTE method. From the results, we can clearly see that the model with review + summary text as input and trained with SVM classifier has accuracy of 91% and performs better than all other models.

This is the result of 5-fold cross validation.
Moving on to the next set of validation, below table(Fig 5.2) represents the results of models trained on pure review text as input converted to vectors with either countvectorizer or TF-IDF vectorizer and feeded to the three classifiers as mentioned in previous set of results.

Review Text	Countvectorizer	Without Balancing	logistic regression	without cross validation	86%
Review Text	Countvectorizer	Without Balancing	Dummyclassifier	without cross validation	63%
Review Text	Tfidfvectorizer	Without Balancing	logistic regression	without cross validation	87%
Review Text	Tfidfvectorizer(bi-gram)	Without Balancing	logistic regression	without cross validation	89%
Review Text	Countvectorizer	Without Balancing	SVM	without cross validation	87%
Review Text	Tfidfvectorizer	Without Balancing	SVM	without cross validation	88%
Review Text	Tfidfvectorizer(bi-gram)	Without Balancing	SVM	without cross validation	92%
Review Text	Countvectorizer	Without Balancing	Random Forest Classifier	without cross validation	86%
Review Text	Tfidfvectorizer	Without Balancing	Random Forest Classifier	without cross validation	87%
Review Text	Tfidfvectorizer(bi-gram)	Without Balancing	Random Forest Classifier	without cross validation	87%

Figure 5.2: Table 2

In this set of validation, again we have the combination of pure review text and SVM classifier as winner with accuracy of 92%. We can also see that the next contender is the logistic regression which almost performed well on this approach with an accuracy of 89%. The vectorizer used in case of the best performing model was TF-IDF vectorizer with bigrams. No cross validation and balancing method was used in any of the cases.
Now, the same set of models as described above was repeated with SMOTE balancing method as shown in fig 5.3.

From the results shown in fig 5.3, the same model of SVM with TF-IDF vectorizer

Review Text	Countvectorizer	Balancing using SMOTE	logistic regression	without cross validation	79%
Review Text	Tfidfvectorizer	Balancing using SMOTE	logistic regression	without cross validation	78%
Review Text	Tfidfvectorizer(ngram)	Balancing using SMOTE	logistic regression	without cross validation	86%
Review Text	Countvectorizer	Balancing using SMOTE	SVM	without cross validation	87%
Review Text	Tfidfvectorizer	Balancing using SMOTE	SVM	without cross validation	85%
Review Text	Tfidfvectorizer(ngram)	Balancing using SMOTE	SVM	without cross validation	90%
Review Text	Countvectorizer	Balancing using SMOTE	Random Forest Classifier	without cross validation	82%
Review Text	Tfidfvectorizer	Balancing using SMOTE	Random Forest Classifier	without cross validation	85%
Review Text	Tfidfvectorizer(ngram)	Balancing using SMOTE	Random Forest Classifier	without cross validation	86%

Figure 5.3: Table 3

performs the best. But the accuracy is decreased to 90% as compared to the previous set of models which were trained on imbalanced dataset.

Next set of models are tried with a slightly different tinge of balancing method. Here a combination of SMOTE and Oversampling is used. From the results shown in fig 5.4, it can be deduced that both Logistic regression and SVM perform equally well with this new approach of balancing.

Next we explore the results of the models trained using the deep learning approach. Here we use 3 different approaches viz. pre-trained Glove embedding approach, RNN(Recurrent neural networks) - GRU and LSTM. From the results displayed in the table below, we can see from Fig 5.5 that the pre-trained model like Glove embedding does not work well on review text. The accuracy decreases to 65% in this case. RNNs are very efficient at catching the semantic relationships between the words on account of their structure to remember values for a longer period of time. LSTMs are seen to outperform the GRU models. With GRU, accuracy of 89% is obtained whereas with LSTM, on an unbalanced dataset, whooping accuracy of 95% is obtained. When LSTM is applied to a balanced dataset, accuracy slightly decreases to 92%, but still is good accuracy. Best accuracy of 96% is obtained when input to the LSTM deep learning model is

Review Text	Countvectorizer	Balancing using SMOTE+Randomoversampler	logistic regression	without cross validation	79%
Review Text	Tfidfvectorizer	Balancing using SMOTE+Randomoversampler	logistic regression	without cross validation	78%
Review Text	Tfidfvectorizer(bi gram)	Balancing using SMOTE+Randomoversampler	logistic regression	without cross validation	86%
Review Text	Countvectorizer	Balancing using SMOTE+Randomoversampler	SVM	without cross validation	80%
Review Text	Tfidfvectorizer	Balancing using SMOTE+Randomoversampler	SVM	without cross validation	78%
Review Text	Tfidfvectorizer(bi gram)	Balancing using SMOTE+Randomoversampler	SVM	without cross validation	86%
Review Text	Countvectorizer	Balancing using SMOTE+Randomoversampler	Random Forest Classifier	without cross validation	77%
Review Text	Tfidfvectorizer	Balancing using SMOTE+Randomoversampler	Random Forest Classifier	without cross validation	79%
Review Text	Tfidfvectorizer(bi gram)	Balancing using SMOTE+Randomoversampler	Random Forest Classifier	without cross validation	76%

Figure 5.4: Table 4

Review text combined with extracted meta data(average word length of the review, character count, word count, sentiment polarity/score extracted using TextBlob) of a balanced dataset. Thus to conclude on the classification models, we can say that deep learning models using LSTMs, outperform the legacy machine learning algorithms. Additionally, the experimentation shows that combining the extracted meta-data like the average word length of the review, character count, word count, sentiment polarity/score extracted using TextBlob help in increasing the accuracy of classification. In the classical machine learning algorithms for classification, SVMs perform the best as compared to Logistic and Random Forest classifier. But Logistic regression is still a good contender in some cases. However in machine learning classifiers, combination of review and summary text only(without the meta data) seems to perform well with SVM.

Let's have a look at the results of the topic modelling obtained using LDA and NMF algorithms. Above figure shows the sample topics obtained from LDA topic modelling algorithm. Similar kind of topics are obtained by NMF algorithm. In this case the

Input type	Balancing method	Method	Accuracy
Review Text	Without Balancing	deep learning- Glove embedding	65%
Review Text	Without Balancing	deep learning- GRU	89%
Review Text	Without Balancing	deep learning- LSTM	95%
Review Text	Balancing	deep learning- LSTM	92%
Review Text + meta data	Balancing	deep learning- LSTM	96%

Figure 5.5: Table 5

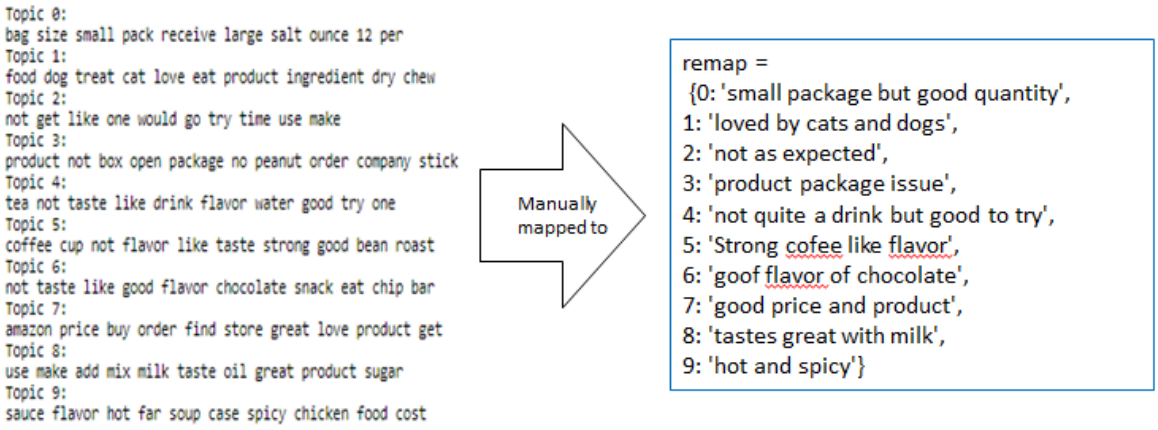

Figure 5.6: Topic Modelling sample output

number of topics is chosen to be 10. As we can see, the topics are random collection of similar words or words which seem to be semantically related. Thus these topics have to be manually interpreted to meaningful topics. For ex. 'Topic 1: food dog treat cat love eat product ingredient dry chew' can be interpreted to 'product loved by cats and dogs'. Similar is the case for other topics obtained. Words in each topic are arranged according to their weightage or score. Coherence score is a performance measurement for topic models. Various models are run for different number of topics as hyperparameters. Coherence score is evaluated for each model and then the model which gives highest score is selected. For LDA model, the number of topics which gave a high coherence score of 0.35 was 14.

Also Python provides a package called pyLDAvis which helps create visualization for LDA models. Following figure shows representation of topics in principal components space. Each topic is represented by a circle where the size of the circle depicts the prevalence of each topic and the distance between the topics shows how distinct the topics are from one another in 2-D representation of multidimensional features. Circles tend to overlap if the words in topics are repeated. The right part of the visualization is the bar representation of the most salient words of any selected topic from the left panel. The bars which are overlaid show the represent topic specific and corpus specific frequency of a word. The left and right parts are linked in such a way that if a topic is selected on the left then the useful terms for that topic get highlighted on the right. Thus this visualization allows the user to examine the topic to term relationship in a compact manner. Finally, once the topics are obtained either through LDA or NMF, they are interpreted manually and each review is then mapped to one of the interpreted topics. Following figure shows some statistic counts of each topic that we can derive once the reviews are mapped to the topics. Such a representation gives an idea what the users are talking about the most in the given dataset.

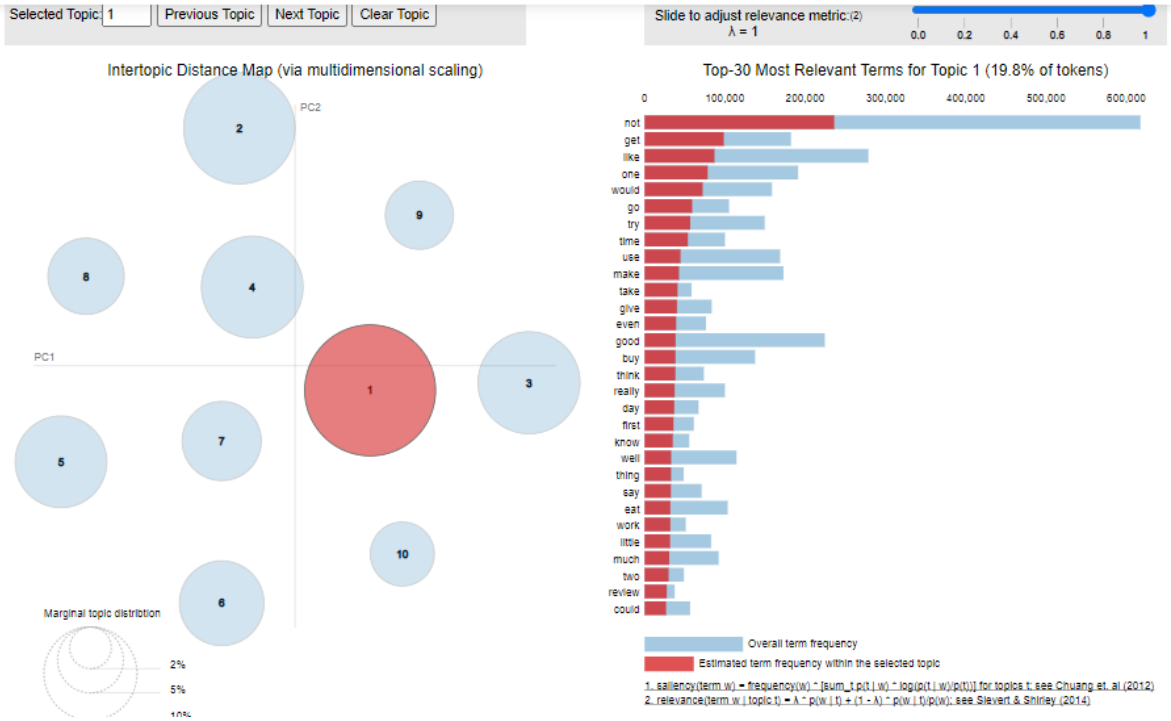

Figure 5.7: Visualization of Topics

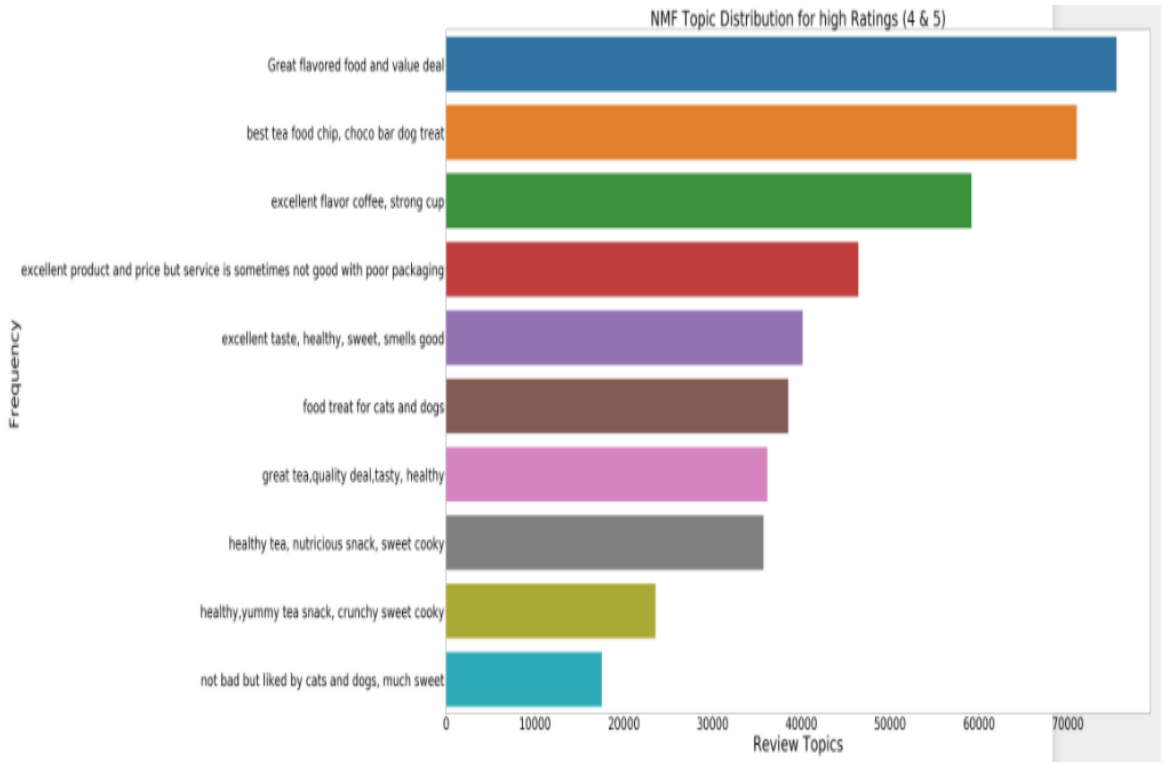

Figure 5.8: Topicwise review count

Chapter 6

Conclusion, Future work and some limitations of this research study

The focus of this study was to build an automatic text classification and topic extraction system and study the effect of the textual meta data on the classification accuracy. The text was pre-processed to reduce the dimensionality and meta data such as average word length, total character length, total number of words, sentiment polarity score was extracted. The cleaned text was then converted into vectors so as to be compatible with input conditions of the machine learning algorithms. Various models were formed and trained based on the combination of: 1. Review text/ Summary text/ Meta data 2. Vectorization methods: Countvectorizer or TF-IDF vectorizer 3. Dataset balancing methods: SMOTE, Oversampling, Undersampling 4. Legacy ML classifiers 5. cross validation. Also, classification of texts was performed using deep learning techniques. In this approach, performance of pre trained models such Glove embedding was compared with the Recurrent Neural Networks(RNNs) such as LSTM(Long Short Term Memory), GRU(Gated Recurrent Unit).

From our experimentation results, we conclude that Support Vector machine performs well in almost all the cases i.e. irrespective of the feature set used. SVM outperformed other legacy classifiers such as Logistic regression and Random Forest Classifier. However, Logistic regression was found to be a strong contender performing just below the SVM in most cases. Additionally, SVM performed well with TF-IDF vectorized features. Accuracy was seen to be dropped in most cases when text was combined

with the extracted meta data and fed to the classifiers. In the Deep learning approach, LSTM and GRU showed better performance and promising results in terms of classification accuracy as compared to the previously mentioned classical machine learning classifiers. Pre-trained models like the Glove Embedding performed not so well as compared to the RNNs and ML classifiers on this dataset. Also, the accuracy of DL models was seen to be increased when meta data was combined with the text. During the execution of the ML and DL models, it was experienced that better pre-processing of the texts helps increasing the classification accuracy. As a second part to this research study, effective summarization of all the reviews has been performed through LDA and NMF topic modelling and appropriate parameters tuning. Effective number of topics which gave the highest coherence score was found to be 14. Thus through topic modelling we get an idea about what specific topics are the users talking about. Last but not the least, through exploratory data analysis, product wise visualisation of the most popular words the users have used for a particular product in their reviews is implemented. From such visualisations, intuitions can be derived about peoples' view about a product or its performance in market.

Every research study comes with its own set of limitations and so does this one. One of the main limitations of this system is that it works well only on a dataset which is labelled. Effective training of the model is possible on this architecture if the dataset is pre-labelled. Thus it is not possible to extract hidden sentiments using this system. Another limitations of this system is that the ML and DL algorithms developed take significant time to run. Main reason for this is that the dimensions of the data get increased due to conversion of text to vectors. Finally, the topics obtained through topic modelling are roughly intuitive because the topics that are generated are just a collection of random words which are semantically correlated. Thus the topics need to be manually interpreted into some meaningful subject. One drawback of manually interpreting topics is that bias can be introduced while interpreting the subject for individual topics.

If future scope of this project is considered, this research can be extended into many directions. Due to the limited features of AWS and no access to the college systems, heavy algorithms could not be implemented due to shortage of system resources. Number of features had to be restricted due to RAM requirements to process high dimensional data. Thus, this research can be extended by increasing the number of features and

combining them with additional extracted meta data like the subjectivity/objectivity, number of noun phrases, number of adjectives, number of verbs, adverbs, considering more combinations of ngrams, etc. This work can be extended to implementing the recommendation model to suggest appropriate items to the users as per the ratings and comments given by them taking into account the biasness correction. Users' likings and dis-likings can be extracted and then combined with the recommendation model to suggest better or similar products to the users. Time series analysis can be performed on this dataset as time stamp is also given along with the reviews. Thus, market performance of the product over a period of time can be studied. Finally, many other efficient ML and DL models can be implemented in this study further giving rise to a more robust algorithm.

Bibliography

[1] Paxcom, "Why is customer sentiment analysis important for your brand." https://paxcom.net/blog/why-is-customer-sentiment-analysis-important-for-your-brand/#:~:text=2)%20Build%20Online%20Reputation%20%E2%80%93%20Sentiment,case%20of%20social%20networking%20sites, May 2019.

[2] U. Rehman, Anwar, K. Malik, Ahmad, B. Raza, and W. Ali, "A hybrid cnn-lstm model for improving accuracy of movie reviews sentiment analysis," *Multimedia Tools and Applications*, vol. 78, no. 1, pp. 1573–7721, 2019.

[3] H. Itai, S. Marc, A, R. Lee, S. Ben, and E. Camila, "Classifying twitter topic-networks using social network analysis," *Social media + society*, vol. 3, no. 1, pp. 1–13, 2017.

[4] B. Aashutosh, P. Ankit, C. Harsh, and G. Kiran, "Amazon review classification and sentiment analysis," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 6, pp. 5107–5110, 2015.

[5] S. ChandraKala1 and C. Sindhu, "Opinion mining and sentiment classification: A survey," vol. 3, no. 7, pp. 420–427, 2012.

[6] Y. Qiang, Z. Ziqiong, and L. Rob, "Sentiment classification of online reviews to travel destinations by supervised machine learning approaches," *Expert Systems with Applications*, vol. 36, pp. 6527—6535, 2009.

[7] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," *EMNLP*, vol. 10, 06 2002.

[8] T. M. Crawford, Michaeland Khoshgoftaar, D. Prusa, Joseph, N. Richter, Aaron, and H. Al, Najada, "Survey of review spam detection using machine learning techniques," *Journal of Big Data*, vol. 2, no. 1, p. 23, 2015.

[9] S. Shojaee, M. A. A. Murad, A. B. Azman, N. M. Sharef, and S. Nadali, "Detecting deceptive reviews using lexical and syntactic features," in *2013 13th International Conference on Intellient Systems Design and Applications*, pp. 53–58, 2013.

[10] J. Lilleberg, Y. Zhu, and Y. Zhang, "Support vector machines and word2vec for text classification with semantic features," pp. 136–140, 07 2015.

[11] V. G and D. Chandrasekaran, "Sentiment analysis and opinion mining: A survey," *Int J Adv Res Comput Sci Technol*, vol. 2, 06 2012.

[12] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," *European conference on machine learning*, vol. 2, no. 6, pp. 137–142, 1998.

[13] J. Zhang and C. Zong, "Deep neural networks in machine translation: An overview," *IEEE Intelligent Systems*, vol. 30, pp. 16–25, 09 2015.

[14] J.Islam and Y.Zhang, "Visual sentiment analysis for social images using transfer learning approach," *2016 IEEE Int. Conf. Big Data Cloud Comput. (BDCloud)*, pp. 124–130, 2016.

[15] A. Severyn and A. Moschitti, "Twitter sentiment analysis with deep convolutional neural networks," *38th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR 15*, pp. 959–962, 2015.

[16] X. Ouyang, P. Zhou, C. H. Li, and L. Liu, "Sentiment analysis using convolutional neural network," *2015 IEEE Int. Conf*, pp. 2359–2364, 2015.

[17] C. Li, B. Xu, G. Wu, S. He, G. Tian, and H. Hao, "Recursive deep learning for sentiment analysis over social data," *2014 IEEE/WIC/ACM Int. Jt. Conf. Web Intell. Intell. Agent Technol.*, vol. 2, pp. 1388–1429, 2014.

[18] W. Li and H. Chen, "Identifying top sellers in underground economy using deep learning-based sentiment analysis," *2014 IEEE Jt. Intell. Secur. Informatics Conf.*, vol. 2, pp. 64–67, 2014.

[19] A. Bagheri, M. Saraee, and F. de Jong, "Adm-lda: An aspect detection model based on topic modelling using the structure of review sentences," *Journal of Information Science*, vol. 40, no. 5, pp. 621–636, 2014.

[20] H. Wallach, "Topic modeling: beyond bag-of-words," *Proceedings of the 23rd international conference on machine learning, ACM 2006*, pp. 977–984, 2006.

[21] M.Steyvers and T.Griffiths, "Probabilistic topic models," *Handbook of Latent Semantic Analysis 2007*, pp. 424—-440, 2007.

[22] X. Wang, A.McCallum, and X.Wei, "Topical n-grams: Phrase and topic discovery," *Proceedings of the seventh IEEE international conference on data mining*, pp. 697—-702, 2007.

[23] M. A.Gruber, Y.Weiss, "Hidden topic markov models," *Proceedings of the international conference on artificial intelligence and statistics*, pp. 163—170, 2007.

[24] Edpresso, "Count vectorizer in python." `https://www.educative.io/edpresso/countvectorizer-in-python`, Aug. 2020.

[25] C. C.Aggarwal, "A survey of text classification algorithms.," in *Mining Text Data*, 2012.

[26] Tony Yiu, "Understanding Random Forest." `https://towardsdatascience.com/understanding-random-forest-58381e0602d2`.

[27] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[28] Colah, "Understanding LSTM Networks." `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[29] Shivam Bansal, "Beginners Guide to Topic Modeling in Python." https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/.

[30] Pattnaik , Priyanka , "Topic Modeling with Non-negative Matrix Factorization(NMF)." https://medium.com/analytics-vidhya/topic-modeling-with-non-negative-matrix-factorization-nmf-3caf3a6bb6da.

[31] A. Varasteh, "Non-negative matrix factorization (nmf) — multiplicative update rules by lee and seung," Mar. 26 2020. Accessed Aug. 21, 2020.

[32] J. McAuley and J. Leskovec, "From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews," *Proceedings of the 22nd international conference on World Wide Web*, pp. 897–908, 2013.

[33] Matthew Mayo , "A General Approach to Preprocessing Text Data." https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html.

[34] Shashank Kapadia , "Evaluate Topic Models: Latent Dirichlet Allocation (LDA)." https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0.

[35] N. Chawla, K. Bowyer, and W. K. LO. Hall, "Smote: Synthetic minority oversampling technique," *Journal of Artificial Intelligence Research*, vol. 16, 2002.

Appendix

Abbreviations: ML - Machine Learning DL - Deep Learning SA - Sentiment Analysis NLP - Natural Language Processing SVM - Support Vector Machine LR - Logistic Regression TF - Term Frequency IDF - Inverse Document Frequency