Performing Sentiment Analysis on Large Email Text Data

Using various Machine Learning algorithms to perform sentiment analysis on email data

A thesis submitted to Auckland University of Technology in partial fulfilment of the requirements for the degree of Masters of Computer Science and Information Science

Supervisor

Parma Nand, Senior Lecturer Computer Science Department, Auckland University of Technology

November 2019

By

Sandeep Sreehari

School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland

Abstract

Companies and organizations all over the world aims to progress and prosper and anyone who wishes so is expected to know about the current progress of the company which can be got from live data. One such live data is found via email data. By analysing email data which comprises chains of conversation between the employees of the company and clients, one can make a judgment as to how well the progress is. But to perform analysis on such large data is tiresome, time consuming and prone to error if done manually. Sentiment analysis which is a domain under Natural Language Processing is a concept which can address this issue. Using Sentiment analysis, we can make such a judgment about the progress of the company or organization. The purpose of this thesis or research work is to bring out the most efficient and best algorithm to perform sentiment analysis on large data set comprising email data with the best precision. This thesis throws light on understanding the basic concepts of sentiment analysis and then showcases a model which performs sentiment analysis on an email data set. Drawbacks of the current model are observed and either an improvement is made to it or a new model is developed to address those drawbacks. Every new model features something new either in terms of handling the data or making use of better classification algorithms and giver better performance values compared to the previous model. The performance is measured in terms of precision, recall and accuracy. In the thesis, an algorithm is demonstrated to show how sentiment analysis is performs where supervised learning is made use of. The next model is built using this model which makes use of a larger email data set. The first model uses a simple K-nearest neighbours classifier to give us the performance measures. The next few models are built to improve the values by using different classifiers and new features such as Named Entity Recognition and Vectorization. In order to achieve greater values, a model was implemented using Artificial Neural Networks and its derivatives like LSTM. Finally, a domain agnostic model built using the concept of bidirectional LSTM gave the best values and this is the model that is presented as the best. The model also has a few features implemented like Word2vec embedding and Dask to improve the efficiency during run time. The literature survey section shows how researching about work conducted by others in the same domain enabled me to come up with the models. The thesis shows an experimental quantitative approach where models are experimented with and a better model is prepared to improve the performance measures. A section is also presented to explain the various concepts, algorithms and formulas used. The thesis concludes by showing the best model to perform sentiment analysis on the large data set and why it is the best. The advantages and strengths of the model are discussed.

Keywords: Sentiment Analysis, Algorithm, Classification, Data set.

Table of contents

Table of Contents

Attestation of Authorship

“I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.”

- Sandeep Sreehari

Acknowledgements

I would first like to thank my supervisor Parma Nand senior lecturer of Computer Science department at Auckland University of Technology. A lot of encouragement was received by Parma Nand right from helping me choose a suitable topic to helping me with the development of my algorithms. He guided me in structuring and articulating my research with a lot of precision. He allowed the research to be my own work but guided me very well whenever I required help.

I would also like to thank Calligo Technologies, a company based in Bangalore where I did an internship of three months where I got to learn the concepts of parallel computation with Dask framework which I have made use of in one of my algorithms. The internship also refreshed my programming skills in Python and more specifically a module called NLTK which is made use of in almost all the algorithms presented.

I would like to thank the Computer Science department for giving me an opportunity to do this research work and I would specially like to thank the student advisor who helped me in being punctual in submitting my thesis and making required progress in time.

Annotations and Details about the Thesis

The entire thesis document contains some short annotations of some words that occur repeatedly.

ANN: Artificial Neural Network

DNN: Deep Neural Network

DL: Deep Learning

RNN: Recurrent Neural Network

Bi-RNN: Bidirectional Recurrent Neural Network

CNN: Convoluted Neural Networks

LSTM: Long Short Term Memory

Bi-LSTM: Bidirectional Short Term Memory

NLP: Natural Language Processing

NER-Tagging: Named Entity Recognition Tagging

KNN: K-Nearest Neighbour

SVM: Support Vector Machine

NLTK: Natural Language Tool Kit

RF: Random Forest

DCT: Decision Trees

LogReg: Logistic Regression

Other key points about the thesis:

* The references are in the form APA-6th

* All the algorithms referred to in the methodology and Results section are present in the appendix as psuedocodes in Python.

* The data set used across all the algorithms in the methodology section is the same. The data set is 200 MB containing email text data.

1. Research Question

This thesis deals with sentiment analysis of text data and the various methods that can be adopted to do so. This takes into account going about different methods and algorithms to address how the sentiment analysis can be performed.

Research Question: To investigate the various machine learning algorithms, to evaluate the algorithms for accuracy and efficiency and optimize the best one for large textual sentiment analysis data. The research will investigate different ways or methods to perform sentiment analysis on large text data.

Issues that form the research question

If we consider a scenario where one wishes to know the quality of something (a product, service, event etc), there exist a way to make a judgment based on the way that something is reviewed. A judgment about the reliability can be made by using existing data in the form of reviews, comments or real time experience from users. We can analyse the way something has been perceived by users and can come up with a method of making a judgment. However, when we consider an object such as a movie or an airline, and we wish to know how reliable the movie or airline is, we study past data about them which could mainly be in the form of user reviews and from this study we can analyse if they are reliable or not. The goal is to not only ease this process of making judgments about the reliability or dependability of something but to also find a very feasible method to do so with maximum precision and accuracy.

In scientific domains, there occur challenges like reliability of substances, formulae or anything related to the domain and there are requirements to know the usability of them. With data sets of text data in the form of comments or chains of email data exchanged between people or scientists, we can use the various concepts of NLP to come up with results, address the various challenges and also optimize the best one of those.

Why the is research question set this way

When we deal with a concept like sentiment analysis which simply means to assess something to be either positive or negative, one can think of a range of different methods depending on what kind of data is at their disposal. The more we study the kind of data we are dealing with, the size of that data, we can then narrow in on particular methods which can be experimented with and if suitable, implemented with. We must keep in mind that this analysis of data is in the perspective of the organization or company and not the user's perspective. Since there is a lot of data available which deals with similar study and methods, we must consider the most suitable method of all and implement a model to perform sentiment analysis on large text data.

How can addressing this research question be useful

When such data which showcases how well a product or service is being perceived by the users is at the disposal of a company or organization, they can come up with ideas to make improvements. This concept has existed ever since mankind got involved in trade but as of today, we have powerful tools and software at our disposal to help us and enable us to address this issue with a lot more precision and efficiency. By putting such software, tools and algorithms to use, we can come up with ways of analysing large text data.

2. Problem Statement

Knowing the performance and progress of one’s company or organization is an integral task of anyone who wishes for their company or organization to progress well. In order to know this, one needs to analyse how the performance of that company or organization has been over a period of time. In order to do this, some valid data is required which can be any data which can tell us about the performance of the company or organization. Once such data is collected, it has to be studied thoroughly in order to make a judgment about the reliability or dependability and whether it will be useful to the organization or not.

Addressing the problem

If the company or organization has products or provides services, then collecting data which is in the form of reviews or comments about the products or services can provide us with data which can be subjected to analysis. So the task is to collect as many data as possible about that product or service we wish to analyse and structure the data collected. This can be found using data sets which can be issued by the company itself or can be found on the internet or is some cases can also be manually prepared by collecting data.

Apart from products or services provided by an organization, even the working ethics of the company can be taken into consideration when we can have access to data which tells us about the interaction between the organization and the customers. Such data can be collected by having access to communication between the former and latter. One such readily accessible data can be found in the exchange of emails between the organization and its customers. Once such data is collected, methods can be considered to solve the issue.

Let us consider an example. Let us assume we wish to know how good an automobile manufacturing company is selling a particular model of a car. We can read reviews provided by the users who have had prior experience with that car. We can use this data (which is in text form) to analyse and judge whether that car will serve the purpose. But the focus is on the company's perspective about how good the car is selling and not the user's perspective. For this, the company can collect user reviews and analyse them. This way, the automobile company can not only get to know how good or bad the car is being sold in the market but can also find ways to improve the sales of that car by addressing the problems which by the users are mentioned in the data.
Emails are a reliable source of such text data for the following reasons:

i. The exchange of text data can be trusted as mostly facts and actual figures are present in the emails.

ii. As most email exchanges in a professional domain are in most cases assumed to be very ethical, the data is in proper text form and not comprise short annotations. This kind of text data can be interpreted much better.

iii. Most of the emails contain date and time of when the email was sent or typed. This data is very useful in increasing the accuracy when concepts of machine learning and NLP are put to task.

So basically, the research question deals with establishing different methods and to come up with the most reliable one of them all.

3. Research methods and architecture of the research.

The research question has been discussed and deals with developing various methods to perform sentiment analysis on large data in text form. Some ways to deal with the question have been discussed along with the research question itself. But in order to address the issue, a suitable research method is required. Now let us look into the research question again and find a way to come up with ideas to answer the question. We are required to come up with some methods to perform sentiment analysis on large text data. Once we have come up with some models, we chose the most suitable one / ones to solve the issue and in turn provide a solution which can also be used by others.

So in short, we need a research methodology that will be based on mathematical calculations, statistics, results and experimentation. The suitable research method to be considered is a quantitative research method. Quantitative research can be described as an investigation done systematically by gathering quantifiable data and performing mathematical, computational and statistical techniques [143]. We will have to consider computational algorithms to analyse the text data and come up with a way to classify the overall data into positive or negative in terms of sentiment. For this, we can make use of several tools, software, mathematical formula and modules. Under a quantitative research, the approach to be selected will have to involve researching about a method to perform analysis on the text data, implement it, analyse the model, come up with the drawbacks and think of ways to address the drawbacks, better the model or develop a better model. This will have to be done by opting an approach which will involve researching and experimenting with the goal of coming up with an even better model than the prior one or improving the existing model in mind.

<u>The research method explained</u>

What is considered for this research is a quantitative research method which will involve experimentation. To debrief the concept, I intend to adopt a method which will be based on computation which make use of algorithms, programs and software, comparing the results derived from various models, comparing the results with the use of tabulation and statistics, then finding ways to either better the existing model or contemplating a new model by researching until a desirable model is developed. We can call this an experimental quantitative research method which is adopted to come up with methods to address the research question.

This method is quantitative because it deals with computational techniques as well as statistical information provided by other sources to perform analysis on data. The results derived by each method are compared using results and algorithms are used to build the models. This method is experimental because it deals with building a model from scratch and developing a better model by researching articles, papers, publications and websites and gathering data to come up with a better model with the intention of addressing the shortcomings of the previous model in mind and presenting the most advanced and suitable model which can also be used by others in the future. An approach is adopted wherein each step research is done to either better the existing model or contemplate a better model.

Reasons for considering this methodology

The goal of the research is to get a step closer to answering the research question with every improvement made to the current model or by developing a new model. In order to provide an understanding of what sentiment analysis is, it is key to start with a very basic model making use of minimal tools and software. We have to keep in mind that sentiment analysis covers a wide range of topics (will be explained in detail in a later part of the thesis) and it requires a lot of research to come up with something better which performs the same task with better ease or efficiency and can bring about other features to better the model. The various aspects of sentiment analysis can be covered by researching more about what others have done and making use of that information to build a better model.

Quantitative research method has a few key characteristics:

1. The research and study to be made is designed prior to the collection of data.
2. Tools, algorithms and computer software is used to collect data and analyse it.
3. Data is usually in numerical or statistical forms.
4. It involves the enumeration of the concepts which are under study.
5. Makes use of experimentation and involves utilizing previously conducted research by others to use so that it can help us in building a model.

The requirements of my research were in agreement with the above mentioned points about experimental quantitative research and as a result, this method was adopted.

4. Literature Survey

There were some papers, articles, books, websites and other sources which gave me an extensive knowledge on various aspects of the thesis such as the algorithms I had to develop, the metrics, concepts of NLP etc. Researching a few such sources of information, extracting the required information and comparing the values was an integral part of this research. This section discusses the various sources of information which offered me knowledge and help to proceed with my research.

In the methodology section we see the basic approach to sentiment analysis using four examples. These algorithms deal with both labelled and unlabelled data sets. In order to proceed with the actual data set which was intended to be used through the thesis, the need to come up with various machine learning algorithms was required to implement sentiment analysis on the text data in the labelled data set. The data set was a large CSV file which was approximately 75 MB in size and comprised email text data. In order to subject the data to training and classification, the data first had to be processed and then a machine learning algorithm had to be used to perform training and classification. In order to come up with suitable algorithms which offer the best precision, recall and accuracy values, research was done using various articles as seen in the Literature Review section. The following were the classification algorithms considered:

1. K-Nearest Neighbours
2. Decision Tree
3. Support Vector Machine
4. Neural Networks
5. Bayesian Belief Network.

Based on the requirements of optimizing the best algorithms for sentiment analysis on email text data, the most likely classification algorithms based on my research were the five algorithms mentioned above. This is the main reason they were chosen.

This comparison enables one to look into each algorithm, see the strengths and weaknesses of each and pick suitable ones.

A paper I researched [56] presents results on the performances of two learning algorithms, Decision Tree classifier and Bayesian classifier algorithms on two text data sets [56]. The authors of this paper came up with approaches to perform text categorization and measure the performances using the two algorithms. The introduction to text categorization is made and the two algorithms are discussed in detail. The algorithms are compared in terms of the metrics and performance. The metrics compared are fallout (or precision) and recall. This paper describes the working of both the algorithms as well. Random Forests and Decision Trees [57] was another paper which I looked into in detail which compared the performances of two algorithms on large data sets. One of the algorithms was RF and the other algorithm was J48 Decision Tree classifier. DTC is the common algorithm studied in detail in both the papers [56, 57]. By comparing the performance values of RF algorithm in both the papers, a way of deciding if the algorithm was to be considered in my research work or not could be used. Apart from this, the two algorithms also describe in detail the working of Bayesian classifier in the first and DTC algorithm in the second. An article [58] gives details of an extensive research conducted which involved the comparison of three different machine learning algorithms and their performances to perform cover classification for image data sets. The three algorithms chosen were RF, SVM and KNN.

Comparison of the methodologies

In [56], the concept of text categorization is described in detail as text categorization is the gist of the paper. The two algorithms Bayesian Classifier and DTC classifier are used for two tasks, one is indexing financial news wire stories for retrieval of documents and the other is extracting data on terrorist incidents from variable text sources. The approaches for text categorization are shows in this paper. They mention two main approaches for construction of text categorization systems. One of the approaches includes many systems and how knowledge engineers define a few layers of conclusions between the words and textual features and output categories and mention rules for mapping between layers and for removing conclusions. The other approach is to use manually categorized text in constructing categories by inductive learning. The two algorithms used by them are then described, the performances of the algorithms are then compared. The evaluation of the performances are described in terms of Fallout, Recall Precision.

Paper [57] compares the performances of DTC algorithm with RF classifier for classifying 20 data sets. The strengths and applications of each of the algorithms are first discussed, then the classification methods are shown. Few of the types of DTC algorithms are describes including the J48 algorithm and then RF classifier is described in detail. The classification performances of the two algorithms are measured using the experimental set-up using large and small data sets. The goal of the comparison is to create a base line which will be useful for classification scenarios and selection of an appropriate model. Results are derived and discussed and the comparison results are show.

In [58], the objective of the research article is to compare the performance measurements of three algorithms on image data sets. The Sentinel-2 system is described [16] and some literature survey is provided. The methodology adopted is a complex process using the image data provided by Sentinel-2. The image data is split into testing and training data. The training samples are subjected to parameter tuning with parameters such as re-sampling approaches. Then the classification algorithms like KNN, SVM and RF are used for classification. The testing data is sent as classified images and finally, the accuracy is measured and the accuracy values are compared. The three classification algorithms and their working are studied and the results are calculated for 8 sub-data sets. The error rate of each algorithm is tabulated for all the 8 sub-data sets. And finally, the performance metrics like Accuracy and Precision are calculated for all three algorithms and compared.

<u>The data sets used</u>

For [56], two data sets are used which make use of different text categorization tasks. The first one was a data set consisting of 21450 stories from the year 1987 which are manually indexed using 135 categories. The training and testing parts of the data set were divided based on a specific date. The data which was present before that data was training set and the data that followed after the date belonged to the testing set. The second data set comprised 1500 documents from US Foreign broadcasting informative service which was used previously for some other NLP task. This data set includes text translated from Spanish language and include newspaper stories, transcripts and other material. Stop words were not removed from the text data.

The authors in [57] got their data sets from UCI Machine Learning repository [60] for classification. Some data is nominal and some are linear. There were 20 data sets used all of which show the number of instances and the number of attributes.

In the article [58], the data sets were in image formats. They were acquired and downloaded from USGS website [61]. These were satellite images taken. The training and testing samples were collected based on the manual interpretation of the Sentinel-2 data and imagery from Google Earth. They used a tool to create about 135 polygons in the data sets.

The algorithms used

There were comparisons made between or among algorithms in the above papers and articles which I researched on. In [56], there were two algorithms which were made use of. The Bayesian classifier and the DTC. The Bayer's classifier was used to come up with an algorithm of their own which they called PropBayes. This uses the Beye's rule to estimate the probability. The formula for the calculation of probability is:

$$P\ (C_j — 1 \mid D) \qquad\qquad \text{--- (1)}$$

The probability that a category Cj is to be assigned to a document D based on the prior probability of a category and the conditional probabilities of particular words in a document which belongs to a category. The DT-min10 approach was based on the DTC learning algorithm. For each category, a decision tree was calculated using recursive algorithm with information gain splitting rule. When there were less than 10 examples that fell at a node, a leaf was forced.

Both the algorithms make use of training examples to estimate conditional probabilities of the occurrence of a feature. The PropBayes can make an estimate of conditional probability of category assigned for a feature but only by assuming a product distribution. D-Tmin10 estimates the conditional probabilities separately for each set of conjunctions of features. This way, the DT-min10 avoids independent assumptions and demands larger training instances.

In the paper [57], there are two algorithms which are made use of. The DTC and RF classifier. Among the decision tree based algorithms, the J48 was chosen. The J48 algorithm is used to generate a decision tree. The WEKA data mining tool was made use of. This, according to the authors is a standard Decision tree algorithm. One of the classification algorithms in data mining is Decision tree induction. In this the classification algorithm is learned inductively to construct a model from previously classified data sets where each data item is defined by the values of the features. The RF classifier is described as a group of un-pruned classification or regression trees derived from random selection of samples of the training data. It is mentioned in [57] that Random Forests generally show better performance as compares to a single tree classifier.

The three algorithms used for classification in [58] are SVM, RF and KNN. As discussed before, the data sets used are image data sets. The radical basis function (RBF) kernel of SVM is shown to be commonly used as a good performer and so the RBF is used to implement the SVM algorithm. Two of the parameters which have to be set are the optimum parameter of cost and the kernel width parameter. The Random forest makes use of two parameters to be set up. One is the number of trees and the other is the number of features in each split. According to [62], larger number of trees provide a stable result of importance. It is also mentioned that making use of more than the required number of trees can be unnecessary but, it does not harm the model [63]. For this study, a range of values for parameters were tested and evaluates to find the optimal Random Forest model for classification. In this, the tree value was taken as 100, 200, 500 and 1000 and mtree value was 1:10 with a step size of 1. The K-nearest neighbors approach is non-parametric which has been used before [64]. The theory behind the KNN is that it finds a group of K samples which are nearest to unknown samples and from these K samples, the class of unknown samples are determined by calculating the mean of the response variables. This is how the ‘K’ value plays an important role in finding the performance of the KNN.

Results and Findings:

The final results obtained as metrics and parameters show the efficiency of the algorithms used. In [56], there were the two algorithms used namely PropBayes and DT-min10 where the former makes use of a Bayesian classifier while the latter uses a DTC. PropBayes estimates a conditional probability of category assignment for a conjunction of feature values but by assuming a product distribution. DT-min10 estimates conditional probabilities separately for every selected set of conjunctions of feature values. The performance measures used were recall and precision. As mentioned, the algorithms were used on two different data sets. On one of the data sets (Reuters data), both the algorithms performed well. The DT-min10 algorithm performed well on the data set at high recall levels and gave about 95% recall value. However, on the other data set (MUC-3), both the algorithms underperformed comparatively. For PropBayes, the performance peaks at around 10 features for the first data set (Reuters data) and 15 features for the other (MUC-3). However, with more features, the performance starts to decline gradually. This is mainly due to the over fitting problem. DT-min10 evaluates the quality of each feature in the context of the features in the tree. On the Reuters data, this algorithm is shown to improve through 90 features and peaks at 4 to 10 features on the smaller MUC-3 data set.

In [57], the classification results of the J48 decision tree and the RF algorithms were compared. Like mentioned in [56], the over fitting problem was likely to occur so in order to eliminate this issue, the accuracy was obtained using 10-fold cross validation which makes use of 90% of the data for training and only 10% for testing. All the 20 data sets, their number of instances and attributes and the correctly and incorrectly classified instances were tabulated. It was shown that the RF classifier gave better results for the same number of attributes and larger data sets. It was also shown that the J48 Decision Tree classifier was better for smaller data sets or when the number of instances are less. The performance metrics measured were Precision, recall and F-measure. Out of the 20 data sets, for one of them, both the algorithms gave very high results. The precision and recall values were 97.7% and 96.9% for RF and J48 respectively. The conclusion they make is that RF classifier gives better, accurate and more precise values in case of larger data sets.

The article [58] compared the performances of KNN, RF classifier and SVM. It was shown from the results obtained from experimentation that SVM gave the most accurate results which were then followed by RF and then KNN. The three highest accuracy values however of all the three classifiers were only slightly different. These three algorithms were tested on 8 sub data sets. The performances of KNN, RF and SVM classifiers were measured on various imbalanced and balanced training samples. SVM gave 96.32% accuracy for one of the imbalanced samples which was the highest. RF classifier gave 94.7% accuracy as its highest and KNN gave 94.57% which was only marginally lower than the accuracy of RF classifier. On the balanced samples however, SVM again outperformed the other two algorithms by giving 95.29% accuracy which was followed by RF classifier which gave 94.59%

and KNN gave 94.1% accuracy. It was shown that all the three algorithms gave accuracy values ranging from 90% to 95% with SVM producing the highest values of accuracy with the least sensitivity to training sample size which were then followed by RF classifier and KNN.

Summary:

The main purpose for researching the above three papers and comparing the derived values was to find suitable classification algorithms to perform training and classification on the data set. Among the various machine learning algorithms present, the need to zero in on only some to experiment on was required. The comparison of some algorithms shown in [25] gave me an idea as to which are the few algorithms I can use to perform classification and training on my data set. Then, the above three papers enables to use four of the algorithms in my algorithms.

The machine learning classification algorithms used in my research gave me decent values of Precision, Recall and Accuracy but there was a requirement for further improvement in the values. Researching papers, articles and websites about Artificial Neural Networks (ANNs) gave me an understanding about the working and structure of ANNs and I came up with ideas of implementing a model using ANNs. Deep learning is a concept of making use of more than one intermittent layer of neural network layers in order to enable the model to train and learn even better. There are some types of neural networks which can be used for deep learning such as Convolution al Neural Networks, Recurrent Neural Networks, Long Short-Term Memory (LSTM) etc. After detailed study, I contemplated an approach to use RNN and LSTM to build a model to improve the values of the parameters. Bidirectional RNNs and more specifically Bi-LSTM was the best concept I could think of based on my research, literature review and study to build a model to get the best results.

The paper [65] deals with LSTM neural network for sentiment classification. The abstract of the paper mentions that RNNs are used more these days to classify text data and are displacing feed-forward networks in doing so. The goal of the paper is to demonstrate the working of LSTM networks and its modifications like bidirectional LSTM and Gated Recurrent unit to classify text data. The superiority of the model over other algorithms for text classification is demonstrated using three data sets. The results are compared with some feed-forward neural networks.

Bidirectional LSTM networks for improved classification and recognition is studied from another paper [66]. In this paper, two experiments are carried out on a speech corpus text data set with bidirectional LSTM and LSTM networks. There are two goals the paper focuses on, the first is to show that Bidirectional LSTM outperforms regular LSTM networks and the second is to demonstrate a hybrid LSTM-HMM system and how it outperforms a traditional HMM system.

Bi-RNNs are studied in detail using a data set in [67]. The goal of the paper is to show how when a regular RNN is extended to a Bi-RNN the network can be trained without the limitation of using input information and how it can give better results than a regular RNN.

Comparison of methodologies:

In the research conducted in [65], dictionaries are made use. There dictionaries are collection of words which describe the surroundings and feelings and their order and context is important. The paper aims to classify entire sentences into groups using RNNs, LSTM, Gated recurrent unit and Bi-LSTM. The paper describes the concept of LSTM and the reasons for the development of LSTM when there were already RNNs available. Bi-LSTM and Gated Recurrent unit networks are explained. The experimental setup makes use of three different data sets. For the experiment, the sentiment labels are divided into three classes ranging from in the scale from 1 to 5 where 1 and 2 are negative, 3 represents neutral comment and 4 and 5 represent positive sentiment. The goal is to classify the content into positive, negative or neutral opinions based on the title. The results obtained from the RNNs and its derivatives are compared with the bag-of-words algorithm. The data was preprocessed in such a way that grammar and punctuation had no influence on the results and only pure word stream was used. The data was trained and results were obtained.

In [66], the paper first describes LSTM network and the hybrid LSTM-HMM recognition model. The experiments were carried out using the TIMIT database [68]. Two experiments were conducted, the first experiment was a frame wise phoneme classification. In this experiment, the frames of speech data were classified into phonemes. The goal was to hand labelled transcriptions with data and the recorded scores were the percentage of frames in the training and test states. The architectures were evaluated using Bi-LSTM, Unidirectional LSTM, bidirectional standard RNN and unidirectional

RNN. For Bi-LSTM they experimented with duration weighted error where the error injected on each frame was scaled by the duration of the present phoneme. The Bi-LSTM hidden layers consisted of 140 blocks of one cell in each and the RNN hidden layers contained 275 units. The second experiment was phoneme recognition in which a HMM was developed with a speech recognition toolkit.

The model and the experiments conducted in [67] do not make use of LSTM or Bi-LSTM as was in the case of [66] and [65]. RNNs are only used in the experiments described in this paper [67]. The experiments conducted in this paper aims to show the performances of ANNs and RNNs over artificial and real-world data sets and to show that RNNs outperform traditional ANNs. Then, a Bi-RNN was also developed which overcame some limitations of RNNs. At first, the concepts of RNNs and bidirectional RNNs were described. Two experiments were conducted on using artificial data and the other using real-world data. For each, the data was described, the experiment setup was shown and results were obtained.

The paper [65] conducts the experiment where the labels are divided into classes representing the opinion or sentiment. Since the aim of the experiment was to classify the opinions based on titles, the results were compared to another model. The aim was to show that the accuracy obtained by using bidirectional LSTM, was greater than that of LSTM. In the paper [66] however, two experiments were conducted and unlike in paper [65], the goal here was to evaluate an architecture using four derivatives of RNNs. But similar to the paper [65], the results showed the high performance of using Bi-LSTM and how Bi-LSTM outperforms regular LSTMs. In this paper, a hybrid

LSTM-HMM model was also developed and even in this, when it was upgraded to a Bi-LSTM-HMM hybrid model, gave better accuracy values. Paper [67] unlike the other two papers does not make use of Bi-LSTM and the experiments are limited to using just Bi-RNNs. There were some limitations observed when RNNs were put to use. In order to overcome these limitations, a model was developed using Bi-RNNs.

So to brief things up, [65] and [66] conducts different types of experiments to show that Bi-LSTMs outperform LSTM models. One experiment was conducted in [65] and two were conducted in [66] so in between the two papers, there were three experiments all of which had different concepts. But in all three, it was shown that Bi-LSTMs give better results when compared to regular LSTMs. Paper [67[did not make use of LSTMs but the results of the experiments showed that Bi-RNNs give better results when compared to regular RNNs.

The data sets used:

Three data sets were used in the experiments conducted in paper [65]. These were:

a) Spam base data set (1999)
b) Farm advertisement (2011)
c) Amazon book reviews data set (2016)

The first data set consists of 13.4% spam messages and the rest were eligible messages [69]. The second data set was collected from text advertisements found in 12 websites which deal with various farm animal topics. 53.3% of the messages were accepted while the rest were rejected [70]. The Amazon book review data set consists of 213335 book reviews for eight different books but the experiment was limited to five books.

The data sets used in the experiments described in paper [66] were from the TIMIT database [71]. The TIMIT comprise sentences of prompted English speech with a full phonetic transcript. It contains a lexicon on 61 phonemes. There were 4620 utterances for training and 1680 for testing. For the experiments conducted, 5% of the training utterances were used as a validation set.

Two experiments were conducted in paper [67] one using an artificial data set and the other using real time values. The artificial data set used for the first experiment, the data is generated by streaming 10000 random numbers between zero and one which is created as a single dimensional input data to the Neural network. The one dimensional output data is derived as the weighted sum of the inputs within 10 frames to the left and 20 frames to the right with respect to the current frame. For the other experiment, real time data was used. The TIMIT phoneme database is used which consists of 6300 sentences spoken by 630 speakers. Two of the sentences are not included in the experiments and the remaining are divided into training and testing sets. The training set consists of 3696 sentences from 462 speakers and the test data set consists of 1344 sentences from 168 speakers.

Results and Findings:

As discussed earlier, three data sets were used in paper [65]. The results obtained on the SPAM data set showed that the model using Gated recurrent unit gave the best accuracy results of 99.945% while Bi-LSTM lagged behind it by a negligible rate and gave 99.834% accuracy. When the Farm advertisement data set was used, Bi-LSTM model gave the best accuracy of 96.017% while the model using Gated recurrent unit gave an accuracy of only 87.521%. When the Amazon data sets was used, Bi-LSTM again had the upper hand in terms of accuracy value and gave 86.4%. Gated recurrent unit model gave 75.821% accuracy.

Paper [66] conducted the experiment of frame wise phoneme classification where various derivatives of RNN were used and their performance values were compared. In this, the findings were such that Bi-LSTM gave the best training and testing values for just 21 epochs. The second best results were from Bi-RNNs. BI-LSTM gave 77.4% and 69.8% accuracies for training and testing sets respectively. Bi-RNNs gave 76.0% and 69.0% for training and testing respectively.

In paper [67], it was shown that Bi-RNNs gave the best results for training and testing sets when compared to the other models on which the experiments were conducted. The unit of measurement was record rate percentage. For training data, Bi-RNNs gave 70.03 % and for testing, it gave 68.53%. These values outperformed the values of other models.

<u>Summary</u>:

All the three papers take into account Neural Networks and more specifically RNNs. Both the papers [65] and [66] conduct various experiments using different data sets to show that Bi-LSTM is liable to give the best performance values compared to many other RNN derived networks. Paper [67] however does not include Bi-LSTM but shows that Bi-RNN has the capacity to give very good results when compared to regular RNNs. LSTM is a derivative of RNN. Researching these papers and comparing various aspects of the papers such as experiments and results, encouraged me to build a model using Bidirectional LSTM.

A paper about Deep Learning (DL) and its various use in NLP was studied [73]. The paper throws light on the recent advancement of deep learning for NLP and also gives us knowledge on its advantages and drawbacks. It is assumed that there are five major tasks in NLP. They are:

1. Classification
2. Matching
3. Translation
4. Structured prediction
5. Sequential decision process

It is shown that DL approaches have outperformed traditional approaches in the first four tasks. Two of the key features that make deep learning powerful for NLP are end to end training and representation learning. It has been observed that deep learning can enhance the performances of the first four tasks. Among the various NLP problems, progress in machine learning translation is remarkable. Machine translation using DL has significantly outperformed traditional statistical machine translations. DL has successfully been applied to image retrieval which involves text to image conversion, a task which was not possible till off late. In this task, query and image are first transformed into vector representations using convolution neural networks (CNNs) and the representations are matched with Deep neural networks (DNNs) and the relevance of the image to the query is calculated [74] .

Some advantages of Deep Learning discussed in the paper:

* DL is employed in natural language dialogue in which, if an utterance is provided, the system generates a response automatically and the model is trained in sequence to sequence learning [75].

* It is good at pattern recognition.

* Cross model processing is possible.

* Learning algorithms is simple.

* Performance is high in many problems.

Apart from the above mentioned, the paper also mentions that the representations of data in various forms using deep learning such as text and images can be learned as vectors.

A few challenges have also been mentioned:

1. Not good for decision making.
2. Difficult to handle long term phenomena.
3. Model is usually difficult to understand
4. Unsupervised learning methods still needs to be developed.

Data in natural language follows a proper law distribution. And hence if the size of the vocabulary increases as the size of the data increases, it means that irrespective of the amount of data there is for training, there always exist cases that the training data cannot cover. By resorting to deep learning alone, this problem is hard to address.

<u>How this paper helped in my research</u>

The task of this paper is very basic. It first gives some information as to what deep learning is and how it has changed things for natural language processing. Then it speaks about the advantages and disadvantages of deep learning.

1. ANNs are the basic most types of Neural networks. It does not enable us to solve a lot of classification and training problems alone. DL solves a lot of drawbacks of regular neural networks and this paper describes the advantages.

2. Some of the advantages included that learning of an algorithm is simple and

it gives high performance. If an algorithm to train a large data set of email text data is used, then there is the requirement of high performance. Deep learning concepts like Deep neural networks appears to be the most feasible choice here.

3. Thanks to the drawbacks mentioned, the drawbacks could be eliminated. The difficulty to handle long term phenomena can be addressed using LSTM.

This paper [76] deals with sentiment analysis on multiple dimensions rather than just using a binary classification technique which may classify data into either positive or negative sentiments.The claim is that it provides more fine grained sentiment analysis. The proposal is about a regional CNN-LSTM based model which consist of two parts used for prediction. This is unlike a regular Convoluted Neural Network (CNN) which considers an entire text as input. In this proposed regional CNN, an individual sentence is considered as a region which divides an input text into many regions. This regional information obtained by this is integrated with LSTM for prediction. The authors propose that by integrating CNN with LSTM the local information within a sentence and long-distance dependency across sentences can be considered in the prediction process.

The multiple dimension is a valence arousal (VA). The approach represents emotional states as continuous numerical values in multiple dimensions. The dimension of valence refers to the degree of positive or negative sentiment while the dimension of arousal represents the degree of calm and excitement. Both the degrees range from 1 to 9. The authors predict that such high arousal texts could help prioritize product reviews. The research has addressed VA recognition on a word-based level as well as a sentence based level. Generally, Convoluted Neural Networks are capable of extracting local information, but they are liable to fail when capturing long distance dependencies. This is where LSTM comes into play. A CNN-LSTM integrated model has been proposed which comprises two parts.

Regional CNN-LSTM Model:
In this model, the word vectors of words are trained from a large corpus using

Word2Vec toolkit. For every text the regional CNN model uses a sentence as a region to divide the text into a number of regions. Useful features can be extracted from each region. This occurs once the word vector passes via neural network layers. These regional features are then sequentially integrated across other regions using LSTM to build a text vector for prediction.
The following Neural Network layers are used to pass the word vectors:
1. Convoluted Layer
2. Max-Pooling layer
3. Sequential Layer.

The values in both the valence as well as arousal dimensions are continuous. The VA prediction requires a regression and hence a linear activation function is used in place of a softmax classifier.

Experimentation:
The proposed CNN-LSTM integration model is evaluated using three methods.

1. Lexicon-based   2. Regression-based   3. Neural-Network based

There were two different data sets used for the experimentation. A standard sentiment treebank and a Chinese VA text dataset. Two lexicon based methods were used for comparison.

1. Weighted Arithmetic Mean (wAM)
2. Weighted Geometric Mean (wGM)

These were used alongside two regression based methods:

1. Average Values Regression (AVR)
2. Maximum Values Regression (MVR).

The valence ratings of English words were taken from the extended ANEW and those of Chinese were taken from Valence Arousal Words (CVAW) lexicons. A CNN, RNN and LSTM were also implemented for comparison.

Performance was evaluated using the following:

Root Mean Square Error (RMSE)

$$RMSE = \sqrt{(\Sigma_{i=1}^{n} (A_i - P_i)^2 / n)} \qquad \text{--- (1)}$$

Mean Absolute Error (MAE)

$$MAE = 1/n\ \Sigma_{i=1}^{n} |A_i - P_i| \qquad \text{--- (2)}$$

Pearson Correlation Coefficient (r)

$$r = 1/n\text{-}1\ \Sigma_{i=1}^{n} ((A_i - \tilde{A})/\sigma A)\ ((P_i - \acute{P})/\sigma P) \qquad \text{--- (3)}$$

In the above formulas:

A_i : Actual Value	n : Number of test samples
P_i : Predicted Value	$\tilde{A}$: Arithmetic mean of A
$\acute{P}$: Arithmetic mean of P	σ : Standard deviation

Results:

Comparative results of the regional CNN-LSTM were put against the results of other methods for VA prediction of text for both English and Chinese corpora. For the lexicon based approach, the wGM outperformed wAM. The regression based methods showed better performance as these methods learnt the correlations between the VA ratings of words and text instead of using the VA ratings directly. The performance of Neural Network based methods improved the performance drastically when the word embedding and deep learning techniques were introduced. The regional CNN-LSTM outperformed the other Neural Network based methods. It is also shown that arousal is difficult to predict as Pearson correlation coefficient of prediction is arousal is lower than that of valence prediction.

How was this paper helpful in my research:

1. While researching about Recurrent Neural Networks, this paper was found to be interesting as it talks about a model which uses LSTM in it. It tells us about the working and advantages of LSTM.

2. The paper gives idea about a higher dimensionality of sentiment classification other than just a binary classification.

3. The strengths of Neural networks and how they are better than traditional machine learning algorithms for learning is identified here.

Some other sources of information I found on the internet which helped me in my research:

This research paper shows how context can better a sentiment analysis model for text data [81].

This paper gives elaborate information about cleaning noisy email text data which is to be subjected to natural language processing [155].

This paper elaborates about how bidirectional LSTM networks are used to parse a graph based dependency. The information gathered by reading this paper was useful to understand how bidirectional LSTM networks can help not only retain memory over a long period of time but also to get more than a single source of memory [168].

5. Introduction to Sentiment Analysis

Sentiment analysis aims to decide the opinion polarity passed on through a range of text data regarding a particular entity [1] This text data can concern anything ranging from objects, reviews, one-liners, comments etc. It deals with the process of categorizing sentiment from textual data and determining what the data signifies and if it a positive sentiment or a negative one [1]. Other ways of referring to Sentiment analysis include Opinion Mining or Subjectivity analysis [86]. It falls under the domain of Machine learning and has picked up pace over the last decade courtesy of the numerous use of social media, blogs, and networking sites [1]. Sentiment analysis refers to classifying the opinion or polarity of the text data into positive, negative or neutral sentiment and at times also calculate the intensity of how positive, negative or neutral it is [87]. There are numerous methods of performing sentiment analysis along with a number of different algorithms that can be used to classify the data based on sentiment [88].

One of the major applications of sentiment analysis is to evaluate the performance of a company’s products or services [89]. This is the application mainly considered in this thesis. In short, by classifying the sentiments or opinion on a product, it gives the company which is dealing with that particular product a better insight of how good the product is in the hands of customers. For instance, if the negative sentiment about the product exceeds the positive ones, then it can be deemed as a bad product for the company. Considering the granularity of the text data, there are different levels in which sentiment analysis can be carried out.

1. Sentence Level [2]
2. Aspect Level [3]
3. Document Level [4]

The thesis goes through the three levels of sentiment analysis and shows how accurate the classifications are. The research included many steps and findings, building a better model over the previous one not only to increase the efficiency or/and accuracy but also to build a more robust model to perform sentiment analysis on larger data sets. Let us go through a very basic approach of Sentiment analysis.

A very Basic Sentiment Analysis Approach

In order to demonstrate a very basic form of sentiment analysis a simple data set of unlabelled text data was made use of. Basically, sentiment analysis comprise three steps, The initialization step where data is collected and preprocessed, the learning step where the data is trained and the evaluation step where data is tested and classified. The first step was to develop a very simple sentiment analysis model which uses the basic method to sequentially classify each text data from a large data set into its respective sentiment according to rules provided.

Let us take a look at Algorithm_1a in the appendix section. This program written using Python3.6 shows a basic rule based sentiment analysis working on a data set of reviews. What this algorithm basically does is, it goes through each line of the data set which contains text data, it makes use of a basic machine learning library called Text blob and classifies each text data into sentiment. In this, the step of pre-processing the data is missing. The text data is subjected to classification as they are. The missing step of processing the data is shown in the next algorithm.

Outline and Working of the algorithm

Refer to Algorithm_1a in the appendix section. The programming language used here is Python3.6 as mentioned. At first, the required libraries and modules are imported. Then the data set is read into a data structure called Data frame which will be explained in detail later. The data set is a large data set of reviews of some online course provided. The module used to create the data frame is Pandas. Pandas is a Python library that consists of data structures and functions which can be made use of to work along with structured data efficiently and in an expressive way and is an important module in python which serves as a good tool for data analysis [5]. The data structure used with Pandas is a data frame which comprise rows and columns of data. Once the data set is loaded into the data frame, we then create a simple list from the data frame so that it makes it easier to iterate through the list. This list just consists of the ‘Review’ column of the data set which consists of the reviews. Now the module Text blob is used to classify the sentiment into positive, negative or neutral using an inbuilt function. Text blob is a library in python which provides API for natural language processing tasks to process textual data [6]. At first, three variables are initialized each to hold the number of positive, negative or neutral sentiments. Then the list containing the reviews is iterated over. Each string or text data is subjected to sentiment classification. This is where the Text blob functionality comes into play. An object ‘Textbook’ under the text blob library is used to create a blob which is in turn stored in the variable ‘blob’. By performing blob.sentiment.polarity, we can get the polarity of the sentiment. Polarity is an object under sentiment of a blob variable which analyses the sentiment of the text and gives a polarity ranging from -1.0 to +1.0 where -1.0 represents absolute negative polarity and +1.0 represents absolute positive polarity. This way, each text data is given a polarity. In the next step, a label is provided which is based on the polarity. The labels range from 1 to 5 where:

1 represents very negative,

2 represents mildly negative

3 represents neutral

4 represents mildly positive and
5 represents very positive.

The *blob.sentiment()* function of text blob uses an inbuilt rule based analyser to give this polarity. In the same step, the positive, negative and neutral scores are also calculated. At the end, the one with the highest score happens to represent the overall sentiment of the data. In this case, it turns out to be a positive sentiment.

Note that the step of pre-processing the data is missing in this algorithm. Now we shall consider another algorithm which does the same as the previous algorithm but where the data is subjected to pre-processing.

Subjecting Processed data for classification:

Let us consider Algorithm_1b from the appendix. The main working is the same as in the case of Algorithm_1a but in this case, the data is first processed and is then subjected to classification using Text blob. When we look at Algorithm_1b, we see the function or method process() which does the processing of data. Data processing is done mainly so that it makes it easier for the classifier to understand the data. Processing of data can include steps like removal of unwanted data such as proper nouns, trimming particular words and others. In the above algorithm, Python's NLTK is used for processing the data. NLTK stands for Natural Language Tool Kit. NLTK is a collection of open source modules and provides tools used for natural language processing [7]. Natural Language Toolkit (NLTK) is a comprehensive library in Python mainly used for NLP functions and text analytics which is adopted in the industry for research and development due to its various advantages [8]. NLTK will be discussed in detail further in the thesis so let us focus more on the algorithm. The process() function loads the data into a data frame, and subjects each text data from the ‘Review’ column of the data frame which contains data about reviews of the course and sends each text data to another function preprocess() which will make use of various functionalities of NLTK. We can see various applications of NLTK like removal of stop-words, word tokenization, stemming and removing of punctuations all of which will be in detail discussed later. This processed text data is sent back and is added into a new column in the data frame. This data frame is then subjected to the same classification steps that are carried out in Algorithm_1a where making use of Text blob, we can derive a polarity for each text data and an overall sentiment can be given.

Comparing the two algorithms:

The only difference between Algorithm_1a and Algorithm_1b is that in the latter, the text data is processed first and then sent for classification but on the other hand, the former lacks this step. This comparison of the two algorithms shows what efficiency processing of the data can provide us with. Let us take a look at image-01 in appendix which show us the time taken for text blob to assign a polarity to each text data, deriving the positive, neutral and negative scores and defining the overall sentiment of the data set. The *image_01* in the appendix section shows the overall sentiment derived using Algorithm_1a and the number of seconds taken for the above three steps mentioned compared with Algorithm_1b. We can see that while Algorithm_1a takes about an average of 45 seconds to assign polarities for each text data, derive the scores and define the overall sentiment, Algorithm_1b takes an average of about 38 seconds which is about 7 seconds faster. The data set used here is about 2.7 MB. The above observation shows the advantage of processing text data before subjecting them to classification.

Now let us consider the same method of using Text blob to analyse a data set of Twitter data.

Analysing Twitter data using Text blob

In the previous examples it was shown how Text blob can be used to analyse text data and classify them into positive, neutral or negative. Now let us consider an example where Twitter data is being analysed. In the next example, a model is shown which accepts parameters and accordingly gives us output values which is an analysis of the data. Tweepy library of python was used in this. Tweepy is a library in Python which can be used to interact with Twitter's API [11]. Tweepy also provides us with access to Twitter API, and so we can get to use data sets of Twitter data [11]. It comes with various functionalities to analyse Twitter data. Then apart from Tweepy, Text blob was used as shown in the previous examples and the same functionalities of Text blob were used to classify the sentiment into positive, neutral or negative.

Reasons for analysing Twitter data:

The process was started with Twitter data or 'Tweets' because each element of text data will be comparatively shorter when compared to very large text data sets like email or documents. Tweets generally comprise one line data and only a few words [90]. However, there were some tweets which consisted of two to three lines of text data in the data set used here. Another reason for considering analysing Twitter data was to show the use and advantages of data pre-processing. Data pre-processing as discussed earlier enables us to derive data which the model can more easily read as compared to the data that is unstructured without being subjected to pre-processing. Since on Twitter data or Tweets, a lot of special characters are used. Apart from these characters, each Tweet consists of an ampersand ('@') and this data will not be required to be subjected to classification as it does not refer to any sentiment. The processing of Twitter data will be discussed later. Twitter data will also contain a lot more emotions and emotes as compared to reviews about a product. These can be considered for deriving the sentiment.

Research:

The algorithm of using Text blob along with Tweepy was developed after researching about Tweepy and its various functionalities. The Tweepy documentation [12] gives a large idea of how the module or library can be used to perform sentiment analysis on Twitter data.

An article very useful in which the authors have made use of Tweepy to perform sentiment analysis based on location which identifies trends towards elections in a country [11] was also useful. In this paper, the authors have come up with a model to analyse the Tweets and also collect the data about its location. Using this, they analyse what political party the user/typist of the Tweet is either for or against and the location of that person. They then plot a map which shows how likely a place in the country is likely to vote for which of the political parties. Along with Tweepy, a Twitter-API related to Geo-location was also made use of. They have also discussed data pre-processing which made use of understanding and analysing emotions.

Method and explanation of algorithm:

Although the algorithm used in this thesis was not as complex as the one mentioned in [11], researching the paper gave an advantage to use Tweepy to perform simple sentiment analysis using Text blob.

Refer to Algorithm_2a. In this, we first encounter four variables once the required modules are imported. There four variables are:

1. consumer_Key
2. consumer_Secret
3. access_Token
4. access_Token_Secret

In order to collect Twitter data and use them, it is necessary to set up a protocol which allows user profiles to access an API. For this, we have to specify the four variables. These are keys and tokens for one to authenticate and make use of the Twitter API. These tokens are generated automatically when we register to Twitter. These four variables are initialized and their respective values are given in string formats. We also have to specify authentication parameters. Two other variables are also initialized for which the user is expected to give in values. The first in the search term. In this the user gives a string data of what term specifically has to be analysed and the second variable is number of search terms which is a numerical value of how many of those terms need to be analysed. Once this is done, we initialize three variables of the positive, negative and neutral values. From here on, the method is very similar to the previous algorithms. First, the Tweets are iterated and for each, Text blob is used to classify it into positive, neutral or negative sentiments and polarity is also calculated. With each variable one of the three variables (positive, neutral or negative values) increases. Finally, the percentage of each of the three values is found and whichever is the highest, the search term corresponds to that sentiment. Image_003 shows a few outputs of the above algorithm.

The above algorithms showed a rule based approach to analyse text data making use of NLTK for processing and Text blob to classify data and derive sentiment. Although is it easy to interpret and generate a rule based classifier, it involves manual work, and has lesser learning capacity [9]. It is also meant to be more time-consuming [10]. In order to overcome the shortcomings of rule based classification, supervised classification was adopted.

Using Machine Learning algorithm to perform Twitter sentiment analysis:

Supervised Learning can be defined as a learning paradigm which deals with the study of how computers learn to use labelled data [13]. So when we consider a labelled data set, we can subject the processed data to classification which makes use of the various algorithms present for classification. We can build a model of our own using these. To demonstrate this, a data set which consists of Twitter data or Tweets are used and are subjected to analysis. The algorithm also shows the pre-processing of the Twitter data and how specifically for Twitter data, special considerations are made to process the data before subjecting them to classification. In the earlier models or algorithms, It is shown how a group of words of text data can be analysed using a simple functionality of the Text blob module and can be classified into groups such as positive, neutral or negative. In this case, and the ones that follow, labelled data is made use of. Which means each text data is already classified and a label is provided which groups that text data into positive, neutral or negative. But what is introduced that is new is that the model is subjected to classification and training. As discussed before, Twitter data is easier to analyse comparatively when we take comment data, reviews of something or large documents. So in this, a simple algorithm which processes the data, and makes use of a simple machine learning algorithm to train the data is made use of.

Reasons for considering this approach:

Basically, classification of data discussed so far is only a basic part of sentiment analysis and does not directly deal with any concept of Machine Learning as there is no training of model present. This example of an algorithm introduces us to how a basic training model can be developed and the information that can be derived out of doing so. Again, the steps of pre-processing the data are shown. This example is also to enable one to understand the process of training data and if he / she can understand this, the research that follows can also be understood with clarity.

Research Conducted:

The algorithm developed can be used for other kinds of data such as comments or reviews of a product. So it is a model that works well across a verity of domains. What varies slightly are some steps in pre-processing the data as Tweets are special text data with certain characters. An article about Twitter sentiment analysis making use of distant supervision [14] was studied in order to come up with this model. The article gives us an idea of how an automated method of classifying sentiments of Twitter messages was made use of. The goal of the above mentioned paper is mainly to help consumers to make a decision of whether or not to buy particular products based on Twitter data about that product. This goal is similar to that of my Research work.

The research of [14] is a result of analysing an article [15] published article and makes use of the same data which consists of movie reviews in the form of Twitter data. The above data has been analysed. The approach followed by the authors of [14] is to eliminate the recognized neutral Tweets and hence considering only the positive and negative Tweets. Constraints such as maximum length of a Tweet, language model and domain are defined. The Machine learning algorithms used for classification are i) Naive Bayes, ii) Maximum Entropy and iii) Support Vector Machine (SVM). The pre-processing of raw data included some steps. Some of them included, removal of only noisy emoticons [16], Removing links, user names from the raw Tweet and removing repeated words. The processed data was then subjected to training using the three classifiers. The derived accuracy using Naive Bayes was 81.0%, using Maximum Entropy was 80.4% and using SVM was 82.9%.

How the above research helped: The primary use of researching the above paper [14] was to get an idea of processing raw Twitter data. The methods followed included removal of noisy (or useless) emotions. It also included removal of repeated words and links from the data. The paper also gives a brief information about three machines learning algorithms namely Naive-Bayes, Maximum Entropy and Support Vector Machine. In order to proceed with the research, It was vital to gain some knowledge about the various Machine Learning algorithms and how they can be used. The working of these algorithms is also shown which helped me.

<u>**Explanation of the Model and methodology**</u>

The model conducted for the above research includes a basic sentiment analysis approach. Let us refer to Algorithm_2b in the appendix section. The program is written using Python3.6 and several modules and libraries have been made use of. We see in the start that required modules are imported. Some of these modules are pandas, which is used to create data frames to store our raw labelled data, sklearn to use machine learning algorithms to train and classify the data, nltk using which we can process the data and several others.

This algorithm or program consists of three different functions or methods. Function ‘process()’ is used to create a data frame, and send each row of data from the data frame to another function ‘pre_process()’ to process the raw data so that it can be easier for training purposes. The method ‘pre_process()’, makes use of several NLTK functions. NLTK, its functions and other modules will be explained in detail further in the thesis. To explain in brief the working of the function, the *word_tokenize* functionality of NLTK which will create a list of words from the text data which is passed as a parameter to the function is used. Then, stop words (Words that are not required. This will also be explained later) are removed. After this step, the sentence is stemmed and finally, punctuations are removed. This is now processed data as unwanted characters such as punctuations and stop words are removed. The raw data is sent back to the function process() and is added to a new row which is created in the data frame. Once every row of data is processed and is added to a new row, the entire data frame is sent to the function ‘tweet_analysis()’. This is where the classification and use of machine learning algorithms for training is used. We can observe that first, the rows of the data frame containing the Reviews and Labels are named and are put into a list for each of them. Then the lists are divided into training and testing purposes. After this, a classification algorithm to classify and train the data is used. This is the K-Nearest-Neighbor algorithm. Using this algorithm, four values have been calculated:

i) Accuracy

ii) Precision

iii) Recall Value

iiii) F-Score

The values of the above parameters are as follows for a selected data set:

i) Accuracy --> 74.65%

ii) Precision --> 67.22%

iii) Recall --> 60.43%

iiii) F-Score --> 63.04%

The Data Set

The data set selected for the above model is a csv file which consists of reviews of an airline. The csv file consists of 14641 columns of data and several other columns. When loaded into a data frame using Pandas, we can see that the various rows in each column consists of the review in text form which are the Tweets, a label which has data in three forms (positive, negative and neutral), a column for the reason of the review and others. What we need to consider are only the reviews and the labels. In this data set the label is in text form of 'positive', 'negative' and 'neutral'. In the data sets used previously for Algorithm_1a and Algorithm_1b consists of labels in numerical forms like 0, -1 and 1. This is a relatively small data set which is 3.5 MB and consists of only Twitter data.

Reasons for presenting this model and algorithm

The above model and Algorithm_2b was presented to show basically how sentiment analysis is performed on Twitter data. Twitter data is comparatively easy to analyse and when we consider data sets which are larger and may consist of a few lines of data, we can use a similar approach. The next few chapters deals with analysing sentiment on a larger data set. Unlike in Algorithm_1a and Algorithm_1b, we are not just analysing each data and classifying them into positive, negative or neutral sentiments. Sentiment analysis is not just about classifying the data but also training the data and calculating the precision and accuracy. For this purpose, the above model is presented.

Note: The Algorithm_2b makes use of K-Nearest-Neighbor algorithm for classification and training. The algorithm along with the reason for considering it will be elaborated later on.

6. Framework of Methodology

The methodology was to follow the research method which is a quantitative analysis involving experimentation. We come up with a model, analyse the model along with its drawbacks and find a way to either improve the model or build a new model either ways intending to address the drawbacks of the previous model. First, the same model which was described in the previous chapter was considered and used but with the email data set which will be used across all the algorithms. In order to derive better results, a model where the pre-processing of raw data is done better was made use of. The concept of vectorization was introduces to give weights for each word and other classifiers were used. The best results were derived when the model implemented using RNNs was used. More specifically when the model using bidirectional LSTM networks was used, the best results were derived.

The basic requirments to go ahead with implementing this methodology is a large email data set and a set of algorithms which are suitable to use to perform sentiment analysis on the data set. We make assumptions such that one classification algorithm can give better values than another but we can only prove it by experimentation.

7. Methodology Section

In this section the main algorithms used, the working and research conducted will be discussed. In the previous chapter / section we have seen the basic models of sentiment analysis to get an understanding of how sentiment analysis works. In this section, we attempt to answer the Research Question by introducing a model, implementing it, noting the drawbacks and either improving the model to give better values or build another model which addresses the drawbacks of the previous one. The section shows a set by step procedure of how the best model was come up with after extensive research and experimentation.

1. Implementing the model on a large email data set

In the previous chapter, a model was demonstrated using which Twitter data was analysed and for classification and training KNN algorithm was used. As discussed, the data set was small labelled data set of Tweets which was about 3.5 MB in size. Here we shall consider the same model for a much larger data set and in this case, the data set consists of chains of emails in text form that are much larger than the ones in the data set used in the previous algorithm for analysing Twitter data. The main reason for using the same model for just a different data set is to show how good the model can work across different domains or different labelled data sets.

Methodology and explanation of the model

Since the same model used for the Twitter sentiment analysis is being used here, the steps are all the same. The only difference in loading a different data set. Refer to Algorithm_3a which is again written using Python3.6. We observe that all the steps are just the same.

The reason for using the same model is, so we can observe how the model can work across various data sets. Apart from this, by comparing the values such as precision, recall and accuracy with the ones in the previous algorithm for Twitter data analysis, we can see if the model can actually perform better or worse with a much larger data set.

Observations

As mentioned, the main reason for comparing the values of the parameters with the ones derived by analysing the Twitter data is to observe how well the model can perform. The data sets used were different. While Algorithm_2b used a small 3.5 MB data set of Tweets, Algorithm_3a used a large email data set. The execution for the latter was more but gave better performance metrics than the former.

Reasons for using K-Nearest-Neighbor classifier

For both Algorithm_2b and Algorithm_3a the machine learning algorithm used was KNN. Among the various algorithms available for training and classification, the KNN classifier was specifically chosen at first simple because it is a non-parametric and a lazy learning algorithm [17]. This algorithm does not make any pre assumptions of data distribution which are helpful in cases where the data sets do not follow mathematical assumptions [17]. It also requires no training data points to generate a model which makes training of the data faster [17]. KNN algorithm is one of the oldest and simplest methods of classification [18]. In a study conducted, it was assumed that KNN always belonged to the same class while examples from different classes were separated by a large margin [18]. In this article, the author mentions that when there is minimal knowledge about distribution of data, KNN happens to be one of the simplest and fundamental methods of classification [19]. All the above mentioned were the reasons I chose KNN algorithm for classification in the above two algorithms.

Note: The working and functioning of KNN algorithms will be explained in detail in a later part of the thesis. Shortly, we shall also see the training and classification of data using other machine learning algorithms, why they were used and how they compare against KNN.

Advantages and Drawbacks

One good thing about the model is that it gave decent results of precision and recall. These values were better than the ones where a smaller data set consisting of Twitter data was used. But still, about 71% precision and 67% recall are not really high values. A better model has to be considered which can either:

i) Have better processed raw data or

ii) Make use of other machine learning classification methods.

This way, we could better the precision and recall values

2. Using Named Entity Recognition to process the data

In most cases, the data sets available which contain data in text form will be in simple English. This text data can be reviews as seen in the data sets used in above mentioned algorithms, comments or Tweets. Authors mention that there are factors that affect the success of a task performed using Machine Learning and the quality of the data is of great importance and if there happens to be irrelevant, unwanted and noisy data then the training of data can be challenging [20]. Preparing data for training is mandatory as unprepared or unclean data can affect a data mining algorithm from training the data properly [21]. The main purpose of cleaning the data is for these reasons. So that during the training phase, the data can be trained better. In the above algorithms we have made use of the NLTK module of Python3.6 [8] to clean the unwanted data. The steps included removal of stop-words, removing punctuations and stemming (All of which will be discussed later in the thesis). Apart from the usage of NER-Tagging, the dropna() function is also used on the Data Frame in order to remove the rows of data will null values.

New method considered

After reading about NER-Tagging and its use in processing of data, using this method in my model was considered. This was added as an additional step in the pre-processing phase where the words recognized as ‘NNP’ or Proper Noun by using the NER-Tagger, were removed. This was based on the assumption made that Proper Nouns are noisy or irrelevant and can be removed in order to better the training of data. In domains such as information retrieval, information extraction and machine translation, Named Entity

Recognition and more particularly classification of proper nouns plays an important role [22]. The KNN method was used here as well for classification and training and then in the next algorithm, three other classifiers were used. Particular entities that have unique context are named entities [173].

Note: The working of NER-tagging will be explained in detail later in the thesis.

Research conducted

Apart from reading about NER-Tagging from various links on the internet and some Python NLTK tutorials, I also experimented around with some text data using NER-Tagging. NER-Tagging enables us to extract data like names, places, organization names events etc. The paper [23] gives a description about how an application was developed which made use of Named Entity Recognition with Maximum entropy to retrieve entity sets from a database [23].

Other articles also showcased the use of Named Entity Recognition where it can be used to get names or organizations or places from large text. Such data can be tagged and can also be removed if one feels that group of tagged entities is not required. This is a functionality of the NLTK module of Python3.6.

Explanation of the model and algorithm

The algorithm that explains this model is shown in Algorithm_3b. This is similar to Algorithm_3a is all the steps except for the pre-processing of raw data which in both the algorithms is shown in the ‘pre_processing()’ function.

In Algorithm_3b, in the data processing method or function, we can observe that a variable ‘tagged’ is used. Another variable ‘namedEnt’ is a list of all the named entities. As every entity is looped over, when a proper noun (tagged as NNP) is found, it is removed. This as mentioned before is based on the assumption that proper nouns are noisy. The steps that follow are the same as Algorithm_3a. In the next method, a few other classifiers will be implemented. The same will later be implemented to Algorithm_3b as well. The same algorithm (Algorithm_3b) is then implemented using three other classifiers (DTC, RF and SVM) and this is shown in Algorithm_3c.

Note: The only difference between Algorithm_3b and Algorithm_3c is that in the latter, there are fours classifiers used as opposed to only one (KNN) in the former.

Additional pre-processing

This algorithm featured a more complex pre-processing module for the raw data. The same pre-processing steps were used in the next two algorithms as well. The pre-processing was based on the text that was present in the data set. This data set comprises email data which are chains of emails. The pre-processing improvements made are:

* In some of the rows of email text data there were dates present such as ‘31/10/2017’. This date was considered unwanted and were removed using NLTK.

* Most of the emails consisted of formalities such as ‘Good morning’, ‘Hello’, ‘Greetings’ etc. These again were noisy or unwanted data and were removed

using a list of such formality texts in string forms. Example - [‘hello’, ‘good morning’, ‘regards’, ‘greetings’].

* A few informal email texts were observed in the data set which consisted of unwanted punctuations. Unwanted punctuations were also assumed to be typos. These were removed.

* NER tagging was used to remove proper nouns. So at the end of the email, mostly there would be a proper noun as a signature. These were removed.

* Some of the rows of data had null data in them. These were dropped using the *dropna()* function.

Observations

After subjecting the same data set to classification and training, the following values of precision, recall, f-value and accuracy were derived.

It can be observed in the results section that with respect to all the parameters, the model which makes use of NER-Tagging gives slightly better values when compared to the model which does not make use of NER-Tagging in the processing of raw data. This is the reason for me to consider either using NER-Tagging.

Drawbacks

The most evident shortcoming is that the precision, recall and f_score were only slightly better when compared to the model which did not make any use of NER-Tagging. The other drawback was that execution time was really high when compared to the other model. The execution of the program took a lot more time when compared to the other model, approximately 4 minutes more. This was mainly because of the large increase in the number of iterations to tag entities. In such a data set, the text data consists of chains of email data which is labelled. If for the current data set, if the execution time is really high, then it is practical to assume that it can take exponentially lot more time for the email data set which is much larger.

3. Introducing other classification algorithms

Based on research, the algorithms using KNN for classification and training were implemented as it was found to have some advantages which makes implementation easy. The other major reasons to consider KNN are mentioned in the previous chapter. Each machine learning classification techniques are used for different purposes and in different scenarios based on the type of data or a method required to classify data [91]. KNN was chosen because of the reasons which are already mentioned in the earlier chapters. It should be noted that all of these classification algorithms when implemented using Python3.6 make use of a module or library called SciKit Learn (Or sklearn). Sklearn is a module in Python which makes use of various machine learning algorithms for training of supervised or unsupervised data and aims to make machine learning easy using high level language [24]. Even the KNN classification was implemented using sklearn module as observed in the algorithms. Every machine learning classification method also comes with its own advantages and shortcomings. As the data set used was labelled and various classification methods could be used to classify data and train them, some other classification algorithms were studied and implemented them. The results of precision, recall, f-score and also accuracy were compared across various machine learning algorithms and the results were observed.

Why is this important

Since the role of this thesis is to analyse sentiment over large data sets, it is important to come up with a model that not only gives the best classification results but also has a high precision and recall value. Initially, in order to demonstrate the working of sentiment analysis over a text data set, KNN classifier was used. KNN comes with its advantages, one of which is that it is easy to be implemented. But one should keep in mind that for a simple data set of labelled text data, other classifiers could derive better or worse values. One way to find this out is to implement them for the same data and calculate the precision and recall values.

Research conducted

A survey of analysing the uses of various methods of machine learning algorithms [25] threw light on the advantages and disadvantages of five different methods. The methods considered were KNN, DTC, SVM, Neural Networks and Bayesian Belief Network. This comparison once again mentioned easy implementation to be one of the advantages of KNN [25]. The method with most advantages mentioned was DTC. It is mentioned that there be no prior knowledge on the domain required to carry out this method, and it can handle numerical and categorical data [15]. Like KNN, it is also easy to be implemented. SVM was said to give better accuracy as compared to other classifiers [25] and can also handle complex non-linear data points better. It was also mentioned that Neural Networks were capable of handling noisy data which was a clear drawback when using NER-tagging.

This gave a brief but informative explanation of a few machine learning classifiers with their uses, pros and cons [26]. In this, SVM classifier is defined as a representation of training data as points in space which are separated by spaces as much as possible. It is mentioned that SVM is effective when it comes too high dimensional spaces and makes use of a subset of training points. Even here, KNN is defined as non-parametric and a lazy algorithm and is not really a learning algorithm and simply classifies based on similarity of input variables [26]. Another classifier is DTC which splits the sample into a number of homogeneous sets called leaves based on the significant differentiators in the input variables. It is mentioned that Decision Tree requires little preparation of data and can handle both numerical and categorical data just like how it is mentioned in [25]. RF classifier is a higher version of DTC. It produces a number of trees and classifies objects based on the vote of all trees. This classifier is shown to handle larger data sets and maintain high accuracy. In [27] it is said that Logistic Regression (LogReg) uses a binary scale to test data points as either zero or one. If in case the value is 0.5 it is classified as 1. LogReg comes with the advantage that it can be used for multi class classification and is quick to train. You can also print out the probability values for each class as well. On the flip side, it takes time to test the data.

LogReg is a model of choice in many classification tasks in fields such as medicine [28]. The authors of [28] summarize the similarities and differences of models and compare them with other machine learning algorithms. They study the performances of using LogReg for classification of a medical study related data and compare the results with the performances of ANNs. The same performances are also observed using other algorithms. The article shows that LogReg and ANN share common features in patter recognition and how ANNs are a generalized form of LogReg. The quality of results obtained using the models depends mainly on three factors:

1. Quality of the data set
2. Care with which the adjustable model parameters are chosen
3. Evaluation criteria to report the results.

Towards the end, the model building process of LogReg and ANNs are analysed and observe which factor has to be considered when making a judgment about results using predictive models. The other algorithms studied for classification results are:

1. SVM
2. KNN
3. DTC

Two studies were performed. One in which they analysed various papers to see what other researchers and/or authors preferred. They noticed that details on model building were more often given for Logistic Regression than ANN and this could be because most probably the forward, backward and stepwise variable selection schemes were implemented using LogReg. They also concluded that LogReg takes more effort on the user's part to achieve the same level of sophistication with ANNs. In the other study, the discriminatory power of both LogReg and ANN were compared. In this, it was observed that both the classification algorithms perform on about the same level but ANNs were more flexible and outperformed LogReg in some cases. Out of 72 papers, the authors of 18% of them chose ANN as the better performer in contrast to just 1% choosing LogReg. 42% of them found no difference. These were with regard to statistical testing. This also shows that ANNs can be used for more sophisticated and larger data sets where they take the advantage and outperform algorithms like LogReg. But, LogReg is preferred where model building is to be made easy.

A study of comparing the performances of Random Forest classifier (RF) with Support vector Machine (SVM) showed the advantages of using RF classifier [28]. Parameters such as accuracy and training time were compared between the two classification algorithms. They noted the differences between the two algorithms the most prominent being that RF classifier is a tree-based classifier and SVM classifier is based on increasing the margin between two classes. To conduct the comparative study, a large data set of mapped data of agricultural was used.

The results showed that RF classifier achieved a slightly better accuracy when compared to SVM. Another advantage of RF classifier was that it requires only two parameters to be set on the other hand, SVM requires a number of user defined parameters. The RF classifier is also capable of handling categorical, unbalanced data and also data with missing values which makes it a better choice.

After researching about these and a few other sources, I implemented a model which makes use of various classifiers and derives the precision and recall values along with f-score.

Explanation of the model and algorithm_3c shows the model where four different machine learning classifiers are used for the same data set. The same four parameters were calculated for each namely Precision, Recall, F-score and Accuracy. Once the data set is loaded, the steps of processing of raw data using NLTK are all the same. Just like how KNN algorithm was used to calculate the values in the previous algorithms, the same way Decision Tree, Logistic Regression and Random Forest classifiers were used.

Reasons for considering and using these algorithms

The objective of this thesis is to analyse sentiment on a large data set of emails. The analysing not only refers to classification of sentiments but also involves training the data. As of now, a single data set is used. But if one requires the training over a verity of data sets, then the scenario is different as different algorithms perform better and has its own strengths and weaknesses when used with different data sets. Since my thesis focuses mainly on sentiment analysis where the text data involves chains of emails exchanged between the employees of a company and their customers, analysis of which of the algorithms can give me the best precision, recall and accuracy when used with the data set had to be made. Another factor to consider is the time taken to classify. After extensive research, It was found that the above three algorithms were better to use for my data set chosen because of their uses and advantages. So this algorithm compares the values of the four algorithms, and we can observe which of them gives the best results. No algorithm is better in general than another [29].

Advantages of Decision Tree:

1. Requires less effort compared to other algorithms for data preparation.
2. Does not require normalization of data.
3. Missing or Null values in data does not affect the process.

Advantages of Logistic Regression:

1. Efficient as it does not require many computational resources.
2. It outputs well calibrated predicted probabilities.

Advantages of Random Forest Classifier:

1. Shown to give some of the best results of prediction for supervised learning algorithms [30].
2. Repeated model training is not required.

The above advantages of the three algorithms were the reasons why I chose to train the model using these classification algorithms.

Observations

We can see the output of executing the algorithm in the results section which shows as output the precision, recall, f-score and accuracy values of the four classification algorithms.

Note: The above algorithms are all implemented using SciKit Learn module of Python3.6 otherwise known as sklearn. In the program/algorithm, we can see that the steps are all the same for each of the classification algorithms. The differences are not evident as sklearn is a high level library to make the implementation easier for the user. The algorithms will all be explained in detail with their advantages, drawbacks, uses and unique features in a later part of the thesis.

Note: The three other algorithms selected are implemented in Algorithm_3b which includes NER-Tagging in pre-processing.

4. Implementing a model based on Vectorization

The previous algorithm showed values of four different classification algorithms for a particular data set. That was intended to compare the different algorithms and to derive the precision and recall values. Any time, when we use a larger data set, we tend to derive better values as the model gets trained over a larger amount of data. But this could also increase the computation and training speed and if the data set has larger and ore noisy data in each row, then there are chances of the accuracy decreasing. These were some of the drawbacks observed in the previous algorithm. In order to overcome these, implementing a vectorization model was considered. Vectorization is basically the process of implementing a loop in such a way that it processes a number of elements of an array simultaneously rather than processing a single element of an array a number of times. It is also used to speed up the execution without the use of loops.

In order to come up with a vectorization model, a model which makes use of the concept of Bag-of-Words was implemented. In this, the CountVectorizer() function of SciKit Learn module was used. The goal of this approach was not just to increase the accuracy, precision and recall values but mainly to improve the way the computation and training were performed. This could be achieved by vectorization as it is a concept where each word in a number of text data groups are represented as the number of occurrences of that word [31]. The main objective of this approach is to quantize each key point into one word and represent each of them by a histogram [31].

Research Conducted

A number of articles are available which shows the working and concept of vectorization and the working of a Bag-of-Words model. A study conducted where sentiment classification was performed using two methods, a bag of words approaches which makes use of Multinomial Naive Bayes algorithm and SVM algorithm and a Redcurrant Neural Network method [32]. The goal of this experiment was to test the role of word order in sentiment classification by comparing the Bag of words approaches where the word order is not present with a Redcurrant Neural Network (making use of LSTM) where the latter can handle sequential data as inputs. The Bag of Words features are created by viewing the documents as an unordered collection of words which are used to classify the data in the document [32]. The data sets used in this experiment include a labelled data set of Amazon food reviews and one on academic reviews. The Bag of words model was trained with Multi Nominal Bayes algorithm with and without Tf-IDF and the same was done with SVM algorithm. The LSTM model was trained using Word2Vec, GloVe and a self initialized vector. The results showed that for both the data sets the LSTM outperformed the Bag of Words model. But training the bag of words models against SVM and MNB algorithms showed that the accuracy and training performance can be improved when compared to the previous algorithm (Algorithm_3c).

Working of a Vectorizer

Let us assume we have two sentences.

Sentence_1 — “the quick brown lazy fox jumped over the lazy dog”

Sentence_2 — “education is what you have if you are not lazy to jump to conclusions and be quick”

When we vectorize the sentences, we get the frequency or occurrence of each word in both the sentences.

So the occurrence of the word “the” is 2.

The occurrence of the word “lazy” is 3.

The occurrences of each word is calculated and is stored as an array or a list.

Explanation of the model and algorithm

Algorithm_3d in the appendix shows the model where vectorization is implemented. The pre-processing of data steps are the same as the previous algorithm. Once the data frame is loaded and the text data is processed by removing noisy unwanted data, it is split into training and testing parts. The CountVectorizer is made use of to perform the vectorization. This basically performs what is mentioned above of vectorizing each word as its number of occurrence. The training and testing lists of text data are vectorized and are sent for training. After this, the precision, recall and f-score values are calculated using the same classification algorithms used in Algorithm_3c but in addition we also have a Multinomial Naive Bayes algorithm. The MNB algorithm was experimented with as I did some research with [32] and this paper gave me some knowledge about the algorithm. The precision, recall, f-score and accuracy values of this vectorization model are shown in the results section.

<u>What is lacking in this method and drawbacks</u>

This approach takes into account four different classification algorithms. We observe that the values of precision, recall and f-score are slightly better than the values derived from the previous algorithm. But values like the accuracy and precision can still be made better or higher. The model makes use of a vectorization method and the concept is similar to using Bag-of-Words. As shown and discussed, the frequency of each word in the text data is taken into consideration. This concept of vectorization comes with a few limitations. First off, this concept does not consider the order of words or its context [33]. Context is something that can offer a lot to the model [34]. Another drawback is the amount of time it takes the program to execute when this method is implemented. It is slightly slower when we implement vectorization.

As discussed, the two major drawbacks are the context of a word is not considered and the execution time is high. Both these drawbacks have to be addressed. So for this, the next thing to consider is an ANNs.

5. Implementing Neural Networks

An Artificial neural network (ANN) is a model which borrow the concept of a biological neural network in the Brain of a human [92]. They work using the concept of pattern matching and are used for classification and regression problems and come with a number of variations for solving numerous problems [34]. ANNs come with its own branch of algorithms such as Multilayer Perception (MLP), Back-Propagation, Hopfielf Network etc [34]. ANNs are new computational tools which when compared to other traditional algorithms have found a wide application in solving a number of real world problems [35]. This journal published [35] gives us a glimpse of the use of ANN in today's problems and mention that ANNs come with remarkable characteristics to process information which are concerned mainly with fault and noise tolerance, learning and generalization capabilities. Neural networks have layers [172]. A neural network has to be designed such that a set of inputs produces desired set of outputs [172].

An article which I came across [36] mentioned a few advantages of ANNs. To name a few:

1. ANNS come with the ability to work with incomplete knowledge.
2. They are fault tolerant. If one or more cells of an ANN happens to get corrupted, it does not prevent it from generating an output.
3. It has a distributed memory.

Apart from the above mentioned, there are several other advantages of ANNs.

In the Appendix-D section, image_06 shows the structure of an ANN.

In image_06, we see three layers, an input layer, hidden layer and an output layer. Each layer has nodes which are represented by circles and the connections between two nodes are represented by lines.

Note: The functioning and working of an ANN will be discussed in the appendix section of the thesis. For now let us focus on the research and implementation of ANNs in the algorithms.

Research conducted

In the previous chapters we observed some of the drawbacks of using machine learning algorithms for classification when we used the Vectorization method and the algorithms before that. In order to address those drawbacks, a model using ANNs was implemented. A research paper [38] gives an overview of ANNs, its working and training. The paper also discusses the applications and advantages of ANNs. They mention that Neural Networks opt a different approach when compared to traditional approaches for problem solving. Since Neural Networks process information in a similar way the human brain does, a computer is liable to be powerful enough to perform tasks which we are not aware of [38]. The paper discusses the working of an ANN wherein there are many ways individual neurons can be clustered together. It also discusses the parts of a Neural Network as shown in the image. Training of an ANN, it’s application in supervised and unsupervised learning, applications and advantages have all been discussed in this paper. Altogether, the information gathered from reading and researching this paper [38] gave me a lot of knowledge about how one can implement a model using ANN to overcome the shortcomings encountered in the previous algorithms and the approaches adopted in them.

Another research report which involved a comparative assessment of three different methods for making a prediction was studied [39]. In this report, the performances of Support Vector Regression (SVR) algorithm, RF and ANNs are compared for predicting and mapping soil that is rich in organic carbon. This task involved more of predicting and not learning. The results showed that SVR was the best to make the predictions but ANNs was behind only marginally. They also mention that it can be bettered with calibration. The task was to make predictions wherein ANN proved to be almost equally good as SVR. But when it comes to learning, ANN definitely outperforms the other two.

6. Deep Neural Networks, Deep Learning and Recurrent Neural Networks

The previous section showed why ANNs were considered and some of the articles, papers, websites and other sources referred which in turn gave me a good understanding of ANNs and why they are useful. However, ANNs do come with their list of drawbacks. Some drawbacks include:

1. Artificial Neural Networks generally require processors with parallel processing capabilities [36]. This is courtesy of the amount of computation required.

2. Many of the times, the structure of an ANN is determined by brute force [36].

3. Over time, it becomes difficult to understand what an ANN has learnt [93].

This led to a requirement of coming up with something that addresses the drawbacks of ANNs.

Deep learning is a machine learning concept that learns features directly from data which can be in various forms [40]. Deep learning enables computational models which are composed of multiple processing layers to learn with multiple levels of abstraction [40]. Authors define Deep learning as a class of machine learning techniques which exploit many layers of non-linear information processing for either supervised or unsupervised feature extraction, transformation and classification [41]. Deep learning is also known as end to end learning because the task is learnt directly from data.

In the previous section, we have seen the basic structure of an ANN with its layers. A DNN is a neural network which generally consists of more than one hidden layer in between the input and output layers [42]. Image_07 in the Appendix-D section shows the structure of a Deep Neural Network.

As we can see, a fundamental difference between an ANN and DNN is the number of hidden layers in between the input and output layers. There are more than just one hidden layer in case of the DNN and this feature enables it to learn directly from the data provided and can learn more efficiently and faster.
Note: We shall discuss the working of a DNN later on in the thesis.

Reasons to consider Deep Learning and DNNs

We have seen some of the drawbacks of ANN. In order to address them, DNN was considered. Here are some of the advantages of DNNs over ANNs:

1. Deep learning outperforms other techniques of machine learning when the data size is large. In case of smaller data size, Machine Learning algorithms are preferred [94].

2. DNNs can handle variations in the data which is learnt [44].

3. Parallel computations can be performed and DNNs can perform better with larger data [95].

4. DNNs are flexible and can be adapted to new problems [95].

So far, we have made use of a few fairly large labelled data sets which consist of email data. Each chunk of text data comprise a few words to a few dozen words and on an average about two to three sentences. The thesis deals with email data which deals with chains of emails exchanged between consumer and seller. The algorithms developed so far deals with classification and training data but not learning. If we wish to achieve not just better accuracy but also improve the output each time the data is trained, we will have to come up with a model that is capable of learning from past data. Deep Learning can provide this strength.

Recurrent Neural Networks and LSTM

RNNs has the capability of remembering past and make decisions influenced by that past data [45]. The working of a RNN deals with taking as their input not just the currant input but also what they have learnt previously in time [46]. One of the main advantages of RNN over ANN is that it can model sequences of data and each sample is assumed to be dependent on the previous sample [96]. Image_05 in the Appendix-D section shows an image of a RNN.

In a regular Neural network of ANN, generally the assumption is that all inputs and outputs are not dependent of each other. But in a case where we wish to predict the next word in a sentence, it is required to know the words that came before it. RNNs addresses this problem by considering the output being dependent on the previous computation [48]. There are forms of RNN which come with their uses. One of these forms is a Bidirectional RNN (Bi-

RNN). A Bi-RNN is a form of RNN that can be trained without the use of input information just to a present future frame [48]. The idea of using RNN is to use a sequential information unlike assuming that all inputs are independent of each other in the case of ANNs [170]. RNNs are similar to feedforward neural networks wherein they can process data from the initial input to the final input but in the case of RNNs, they make use of back-propagation [171].

Long Short Term Memory (LSTM) come with the feature of remembering information for a long period of time. LSTM has a complex structure and consists of many gates in each layer. LSTM comes under the category of RNN and is a more advanced and sophisticated form of RNN.

Using these concepts to build a model

In this chapter we have seen the concepts of ANNs, deep learning, DNNs, RNNs and LSTM. The next thing done is to use these concepts to build a simple model to demonstrate the use and working of a Neural Network model for classification and learning.

Refer to Algorithm_IMDB in the appendix which shows the working of a the model. This model makes use of LSTM layers to train an IMDB movie review data set. This model makes use of RNNs in the form of LSTM layers. Keras is used here to build the neural network layers. Keras is a high level API provided by Python which is a framework for Tensor flow and enables more efficient experimentation on deep neural networks [49]. Note that this program is only to show the reader how neural networks can be used to perform sentiment analysis.

Explanation of the algorithm

Just like the other algorithms which did not include neural networks for training, a data set will have to be loaded to perform sentiment analysis. Keras has its set of inbuilt data sets. This is provided by the *keras.datasets* module. From this module, the IMDB data set is downloaded. This data set consists of a movie review provided by IMBD [50]. Parameters are set to load the data set in a desirable form such as maximum words. The next step includes splitting the data set into training and testing purposes. Keras uses a function add a new neural networking layer. Here we see three layers being added. The first is an embedding layer, the second is a LSTM layer with a dropout value of 0.2 and the third layer is a Dense layer. By using the model.compile() function, the layers are compiled. Like in the previous algorithms, model.fit is used to train the data and finally the parameters like accuracy are calculated.

Note: The layers of Keras are discussed in the appendix section.

7. Implementing a model based on Recurrent Neural Networks and LSTM Networks

The previous algorithms were based on using SciKit learn classification algorithms to train the model. Sentiment analysis was performed using those algorithms and the precision, recall and accuracies were compared. The algorithms came with some drawbacks and with the intention of eliminating those drawbacks and come up with a better model, Neural Networks were considered. The research question is about coming up with methods to perform sentiment analysis on large email text data and so far we have been coming up with a better model which addresses the drawbacks observed in the model prior to it. Since we observed some limitations of using the vectorization concept in Algorithm_3d, research about Neural Networks was conducted and implemented in a model.

Reasons for considering this approach

While the classification algorithms used so far all machine learning algorithms, Neural Networks use their own set of algorithms which enables the machine to learn. The advantages of Neural Networks were discussed in the previous chapter. Now let us take a look at some of the advantages of Neural Networks over other machine learning classification algorithms:

1. Neural networks and more specifically Deep Learning enables the machine to learn based on the context of text with time.

2. ANNs make good use of parallel computation [95].

3. Large data can be handled better with ANNs and DNNs [94].

It is important to note that the concepts of Neural Networks are different from Machine learning. Both have their uses in various applied areas. A Neural Network is a subset of machine learning. Machine learning algorithms at some point is liable to give inaccurate values but a Neural Network algorithm is capable to make accurate predictions.

Research Conducted

Using deep learning for text feature extraction has been demonstrated in this article [51]. Here the authors mention that traditional methods of feature extraction require more features to be done manually. They mention the requirement of something more efficient. It is mentioned that deep learning comes with the feature of acquiring new feature representation from training data. Their article mainly focuses on text mining and deep learning does not fail to make things easier. A major feature that differentiates deep learning from conventional methods is that the former automatically learns features from large data and does not rely on prior knowledge of designers. Feature extraction of text which extracts text information to represent a text message is a basis of large number of text processing [52]. Their thesis [51] deals with summarizing a lot of literature to present a text feature extraction method and develop a model based on it.

Another paper examined showed the introduction of a Recurrent Convolution al Neural Network model for classification of text [53]. Their model involves the application of a recurrent structure to capture contextual information when learning word representations. They also create a Deep Learning layer called

Max Pooling layer which can automatically judge which words play key roles in text classification. Four different data sets were used to conduct the experiment. The results of the experiment showed that their model outperforms other models mainly on document level data sets. The pre-processing of the data sets were done using Stanford Tokenizer [54] for the English documents to obtain tokens. Six different methods were chosen for comparison. The experimental results showed that the Neural network methods which included Recurrent Convolutional Neural Network (RCNN) outperformed the other methods. When the RCNN model was compared to Convolution Neural Network model, RCNN gave better results.

Explanation of the model and algorithm

This is a Neural Network based model. It was designed keeping in mind the various advantages provided by Neural Networks. The function *process_reviews()* is used to process the reviews which happen to be the column in the data set comprising text data. From this function, the text data is sent to *clean_reviews()* function where NLTK functions are used to remove unwanted and noisy text. The function *process_labels()* is to reduce the number of classes to three instead of five. The function *process()* deals with the rest of the tasks. Pandas is used to create a data frame which stores the data of the data set. Then we come across the *load()* function. Keras modules are imported and the data set is split into training and testing parts. We can then see Neural Network layers being added. There is an embedding layer followed by two LSTM layers. There is a Dense layer and finally an Activation layer. Finally, the Accuracy, Precision, Recall and F-score are calculated.

Let us get back to the line which contains the *load()* function. Here, a.model file saves the previously learnt information. Each time the program is executed, it loads this information, trains it and saves it back to the.model file. You can see the saving happening at the end of the *process()* function represented by the *save()* function. This was, each time the program is executed, it learns from the previous model.

8. Implementing Word2Vec Embedding

The word2vec is a group of models which are used to produce word embedding [55]. Word2Vec is basically a Neural Network model that have two layers and make up contexts of words. This is a context based learning. It takes a large corpus of text as inputs and produces a vector space [55]. Word2Vec model is used for learning vector representations of words which after the learned vectors are fed into a model are used to generate predictions [55].

Implementing Word2Vec in the algorithm

Neural_Net_Word2vec in the appendix shows the same model as Neural_Net_01 algorithm but with word2vec embedding. Every other step is similar except for the addition of those few lines. We first see five parameters with values. Then a model is defined which produce word embeddings. The trained model is then saved as a.model file. The same number of epochs are used for both the algorithms to train the training data.

The results section shows the readings of Precision, Recall, Accuracy and F-score of the model with Word2Vec embedding. We can observe that the values are slightly better when Word2Vec is implemented.

9. Implementing a Bidirectional LSTM Model

As the research question talks about the different methods we can come up with to analyse large text data, so far we have been analysing text data consisting of email data with simple data pre-processing steps. Now we will use a few complex pre-processing steps to clean the email data. This data set consists of chains of emails between company and customers. The problem statement has been discussed before. In this section, we shall only see the working of the model. For this method, machine learning algorithms would prove to be very inefficient. Deep Learning is used here because this data has to be analysed with regard to context and has to be learnt just like the previous algorithm discussed.

A few papers, articles and web pages were studied in detail which gave me knowledge on Bidirectional RNNs and Bidirectional LSTM. Reading a paper which deals with sentiment analysis on multiple dimensions in order to classify data into either positive or negative gave me an understanding about the uses of LSTM. The literature review section of the thesis shows some papers, articles, websites and other sources which were researched and which enables me to build this model.

Explanation of algorithm

Bi-LSTM shows us the model. This is a very large algorithm so only a step-by-step process is shown in the appendix instead of the python file. If we refer to the algorithm, the large email data set is read into a pandas data frame. A lot of functions are present which are used to pre process the data but in this case, the pre-processing does not only include the use of NLTK functions but also other ways like stripping the large chunk of text data. Email data is very large when compared to smaller text data like Tweets. Especially when the data includes chains of emails exchanged between two or more people. Three main steps were included to pre process the data such that some unwanted text is removed.

In the algorithm, we see that the overall positive, negative and neutral scores are calculated. For these, we have three variables initialize. Like the previous algorithm, parameters like top words, maximum length and nb_classes are provided. The data frame is split into Reviews and Labels sections and each of those are split into testing and training parts. Then we can see the neural network layers created using Keras. After this, the model.fit performs the training of the data and then finally, values of precision, accuracy and recall are calculated.

Pre-processing the raw data

1. The algorithm is a context based model. Stop words cannot be disregarded. So a custom list of stop words were made use of.

2. Smaller rows of data which comprised only small words were assumed to make less sense and were removed.

3. A dictionary is created which contains short annotations of words as keys and their respective formal words are stored as items. For example, “aren’t” is represented as “are not”.

Then, keras modules are imported and a previously trained model is loaded. Keras layers are created, the data set is split into training data and testing data. Finally, the precision, recall, f_score and accuracy are calculated.

The precision, accuracy and recall values are slightly lower when compared to the previous model. But this model has the following advantages over the previous one:

1. It can handle very large data
2. It can learn better thanks to the usage of Bidirectional LSTM.
3. Data is preprocessed much better.

10. Making the model Domain Agnostic

A domain agnostic model is one which can learn across various types of data. The algorithms discussed so far are not domain agnostic. So as the data keeps getting trained, the machine learns but it is specific to one domain. Making a domain agnostic model is to train the model in such a way that it will be able to learn and recognize data (text data in this case) from different domains. This can be very helpful in industries today.

Method: By collecting a number of data sets all of which contain text data about different domains, we run a Python script which read every one of the data sets into data frames, and then a model is trained on all the data sets and stored. The next time the script runs, it loads the trained data from the.model file and repeats the process. This trains the model to be domain agnostic.

11. Implementing Dask

Dask is an open source flexible library for parallel computing in Python [175]. Dask will be explained in detail later. In this small section we shall see how dask was used in Algorithm_email_SA to make it slightly more efficient. In Algorithm_email_SA, the data frame was loaded using Pandas just like in every other algorithm in the entire research work. In this algorithm, since the data set is very large, pandas takes a long time to load it. For this reason, dask was considered. Since it finds its application in the domain of parallel computing, Dask comes with its own data frame. One of the most prominent differences between a pandas data frame and a Dask data frame is that a Dask data frame cannot be worked on. That means, we cannot make any changes or edit the data frame. But this is not the case with a pandas data frame. But loading the data set into a Dask data frame takes relatively less amount of time. So this is what was implemented:

The data set was loaded into a Dask data frame. Then using the *.compute()* function, the Dask data frame was converted to a Pandas data frame. The conversion did not take a long time and so, the loading of data set into the data frame took lesser time and then after conversion, the data frame could be implemented.

The same concept could be applied to the data frames in the other algorithms as well but Dask has the upper hand in terms of speed and efficiency only when the data to be loaded is very large.

8. Results and Findings

The previous chapter of methodology shows the various algorithms and their working. We started with a model which made use of the KNN classification algorithm to train and classify the data from the data set. As the research question talks about evaluating various machine learning algorithms and fine tune the best ones for large text data. Each algorithm developed either came with some drawbacks or there was a requirement to better the metric values so a better model was developed. The research method is a quantitative research based on experimentation. So in each step, we experiment based on the values derived from the previous model and either implement a new model or better the existing one to bring out better values.

Data Set

The data set which is used in all the algorithms is an email data set which comprises exchange of chains of emails between the employees of a company and their clients. The data set is a CSV file which is about 200 MB in size. This is a labelled data set which consists of a few columns two of which are Email Body and Labels. The labels of the data set represent classes. There are 5 classes which range from a numerical value of 1 to 5. 1 represents very negative, 2 represents negative, 3 represents neutral, 4 represents slightly positive and 5 represents very positive. In order to perform binary classification on the labels, in each algorithm developed by me, the 5 classes were reduced to 2 classes. Based on the research question, The 5-class labels have been reduced to a binary class. In each algorithm, the labels 1 and 2 are converted to 0. On the other hand, labels 4 and 5 are converted to 1. Wherever the label is 3, the respective text data in the data frame is considered and a textblob function is used to see if it is more towards the positive or the negative side. Based on this, it is either put into 0 or 1. Hence the 5 classes are reduced to 2 classes. The large part of the email data fell into classes 3, 4 or 5 which made the data in the data set signify slightly positive sentiment. There are a few lines of unwanted data or null data in both the review and the label columns. These were removed so the operations that can be performed on the data frame were easier and the precision value would also increase.

One of the issues with this data set was the large amount of uncleaned data. Many steps had to be taken to clean the data which consumes time even while pre-processing the data. Each row of the data set had large text data which comprised the body of the email. Most of them had formalities such as “Good Morning” or “Hello sir” and did not provide any use for the sentiment analysis. All of those had to be removed. Moreover, some of the rows of data also consisted of dates and signatures which are considered noisy data and had to be removed.

Results:

At first, the model mentioned above is implemented with three other classifiers using SciKit Learn module. The performance values are observed with all four algorithms. The model made better in order to improve the performance values. To achieve this, Named Entity Recognition tagging is used in the step where data is subjected to pre-processing in order to become more clean. Along with this, null values are removed from the data set. The values of the four classifiers are measured using this algorithm. To improve the performance values even further, a model based on vectorization was considered and implemented. This model uses a concept similar to bag-of-words approach and provides weight for each chunk of text like word. This was shown to improve the values even further. Among the four classifiers used, SVM classifier gave the best precision and recall values. The need to improve the performance values even further gave rise to the implementation of a model based on Artificial Neural Networks and more specifically DNNs. A model was built using Keras where network layers like LSTM layer were implemented. The model gave slightly better precision value compared to the model based on vectorization using SVM classifier but gave much better recall value when compared to the latter. To better this model, Word2Vec embedding was implemented in the same model. This showed slightly better values. A final model was brought out which made use of Bidirectional LSTM layers, more complicated pre-processing of raw data, a domain agnostic model was implemented. This gave the best values and was chosen as the algorithm to be presented in the thesis.

The metric values are tabulated and the results are shown. Table_01 shows the tabulated results of Algorithm_3b when classified using the four chosen classification algorithms.

Metrics -->	Precision	Recall	F - Measure	Accuracy
Algorithms				
KNN	71.801 %	67.796 %	69.546 %	80.363 %
Decision Tree	65.311 %	65.123 %	65.216 %	90.211 %
Random Forest	65.672 %	65.352 %	65.509 %	91.322 %
SVM	89.204 %	74.528 %	81.414 %	88.280 %

Table_01: Results of Algorithm_3b when classified using the four algorithms

In the above table we see the precision, recall, f-measure and accuracy values of all the four algorithms. It is clear that SVM derives better results of precision and recall when compared to the other three algorithms. RF classifier gives the best accuracy results among the four classifiers but the important metrics here are precision and recall and hence F-measure. In order make an attempt to better the values, a model that makes use of NER-Tagging and also the *dropna()* function of Pandas were made use of which removes text data with null values or missing values. Table_02 shows the results of Algorithm_3c which uses NER-Tagging and *dropna()* function.

Metrics -->	Precision	Recall	F-Measure	Accuracy
Algorithms				
KNN	72.317 %	68.567 %	70.201 %	81.927 %
Decision Tree	65.938 %	65.768 %	65.853 %	90.774 %
Random Forest	66.086 %	65.992 %	65.992 %	91.829 %
SVM	89.484 %	74.873 %	81.529 %	88.866 %

Table_02: Results of Algorithm_3c when classified using the four algorithms

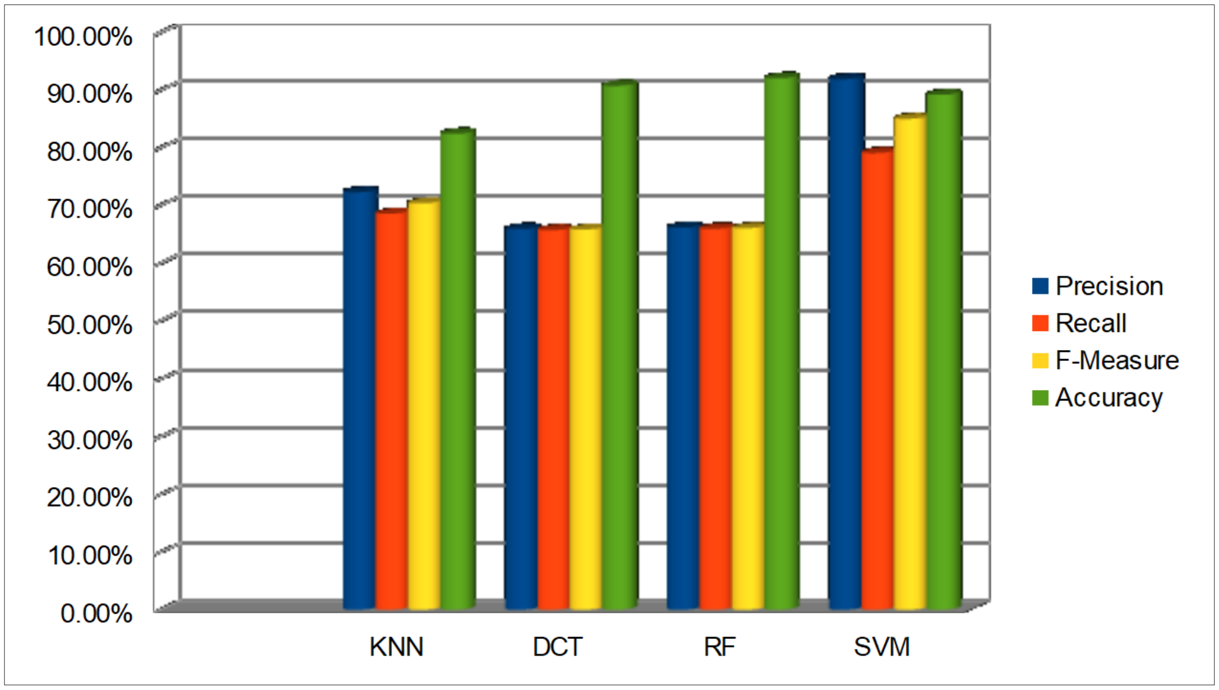
In table_02, we can see that the metric values are slightly better than the ones in table_01. Both the accuracy and the precision recall values are slightly higher. This is because of the use of NER-Tagging which was used to identify proper nouns, organization names and events from the text data set and clean the data. This made the training steps easier. Moreover, the dropna() function which was absent in Algorithm_3b was made use of here. This function drops or removes the lines in the data set with missing values. NER-Tagging was made use of in the pre-processing step in the algorithm. The steps are discussed in the methodology section. Because of cleaner data being subjected to training and classification, the precision and recall values are slightly better in this algorithm and also because of missing values being removed. In table_02 similar to table_01, we see that SVM is the algorithm which outperforms the other three algorithms in terms of precision and recall values. The precision and recall values are <u>89.204%</u> and <u>74.528%</u> respectively for Algorithm_3b and for Algorithm_3c, the precision and recall values are <u>89.484%</u> and <u>74.873%</u> respectively. There is only a slight improvement in the values in Algorithm_3b when compared to the values in Algorithm_3c. Table_03 shows the tabulated metric values of the four classifiers when used with Algorithm_3d which uses the concept of vectorization.

Metrics -->	Precision	Recall	F-Measure	Accuracy
Algorithms				
KNN	72.329 %	68.618 %	70.426 %	82.463 %
Decision Tree	65.968 %	65.787 %	65.877 %	90.792 %
Random Forest	66.188 %	65.992 %	66.089 %	92.083 %
SVM	91.917 %	79.116 %	85.052 %	89.218 %

Table_03: Results of Algorithm_3d when classified using the four algorithms

The table above shows the values derived from Algorithm_3d in the appendix section. This model makes use of vectorization concept similar to a bag-of-words approach.

The graph below compares the parameters of various classification algorithms



Reasons why Algorithm_3d gives better values than the other two:

The vectorizer is defined using the *CountVectorizer*. These makes use of numpy arrays and converts each element into a vector. Vectors are created which have the dimensionality of the size of the vocabulary so if the text data contains that vocabulary (or word), then the count of that dimension increases. If there are no occurrences then a ‘0’ is put in that place instead. We will have a large list of vectors, and we use them on the text data. The value of that vector is represented as the weight for that vocabulary (or word). Since there is a weight for each word, the algorithm will process the ones with higher weights with even more accuracy. This is the reason why better values are got when compared to the previous algorithms.

Algorithm_3d was built on top of Algorithm_3c. So the former was an implementation of vectorization with NER-tagging concept in the data processing steps. Algorithm_3c is built on top of Algorithm_3b. So each is an improved version of the earlier one and for all three of the algorithms, SVM classifier gave the best results. Considering this, Algorithm_3d implemented with SVM classifier was chosen as the best model till present. This had the following values:

Precision: 90.917%

Recall: 79.116%

F-Measure: 84.607%

Accuracy: 89.218%

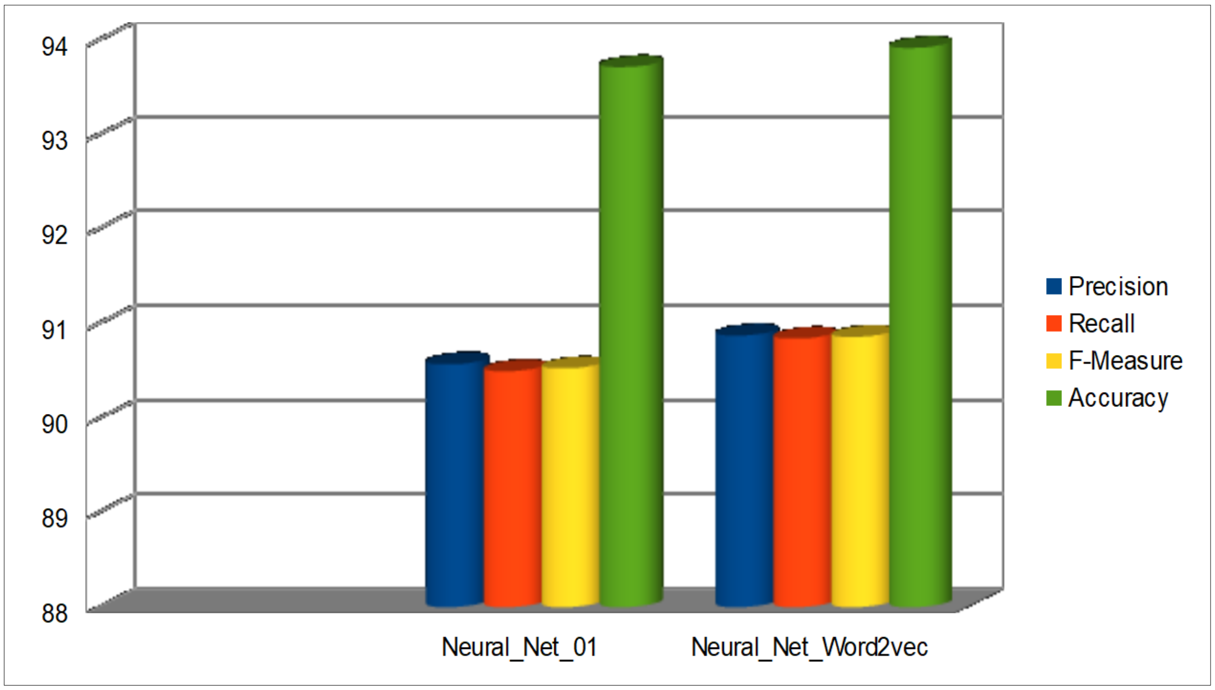
Reading and researching about the strengths and versatility of ANNs and DNNs encouraged experimenting with a model based on the above two concepts and come up with an algorithm. Keras was used to implement the model and some neural network layers were used. LSTM is known as Long Short Term Memory and is a type of Recurrent Neural Network. This was the main network layer used in the next algorithm. So Algorithm Neural_Net_01 was implemented using the above mentioned concepts. This algorithm gave better F-Measure value and accuracy value when compared to Algorithm_3d. But there was a need to make the model context based and exploit the resources provided by Neural networks and deep neural networks to further improve the performance values. For this, the same model (Neural_Net_01) was given an addition of Word2vec embedding. The working and concept of Word2vec is explained in the next chapter. The addition of Word2vec embedding improved the performance values only slightly.

Metrics -->	Precision	Recall	F-Measure	Accuracy
Neural_Net_01	90.571 %	90.487 %	90.528 %	93.714 %
Neural_Net_Word2vec	90.875 %	90.843 %	90.858 %	93.917 %

Table_04: Comparison of performance values between Algorithms Neural_Net_01 and Neural_Net_Word2vec

In Table_04 we see that the performance values of Neural_Net_Word2vec are better than Neural_Net_01. It is to be noted that the latter is built on top of the former. The latter uses the same model as the former does but with the implementation of Word2Vec embedding.

The graph below compares the parameters of the two algorithms involving Neural Networks. It is evident as to why using Word2Vec embedding is better.



<u>Why does the model with Word2Vec embedding give better values than the model without it</u>:

Word2Vec embedding works in such a way that it takes the context of every word that is present in the input text data and attempts to predict the word which corresponds to the context of the word. It makes use of vectorization concept and matches a word to its context. This is the reason why it gives slightly better values.

In table_04, since among the two algorithms, the one with Word2vec embedding gives slightly better values, we consider it. The performance values are:

Precision: 90.875%

Recall: 90.843%

F-Measure: 90.858%

Accuracy: 93.917%

Researching about Bi-RNNs and more specifically bidirectional LSTM networks enabled me to consider implementing another model which makes use of bidirectional LSTM layers. Although Word2Vec embedding was used in this model as well, it was not based completely on the previous model (Neural_Net_Word2vec). This model made used of a pre-trained model in order to make it domain agnostic, was implemented with bidirectional LSTM network layers, used a different set of data pre-processing steps, creating a custom set of list of words to be considered and not considered as clean text data and to improve the speed of execution used Dask framework. Bi-

LSTM_Algorithm presents this model. The implementation of the model using the above mentioned concepts gave the following performance values:

Precision: 92.715%
Recall: 92.715%
F-Measure: 92.715%
Accuracy: 93.945%

We now have three algorithms which make use of different concepts. The three algorithms are:

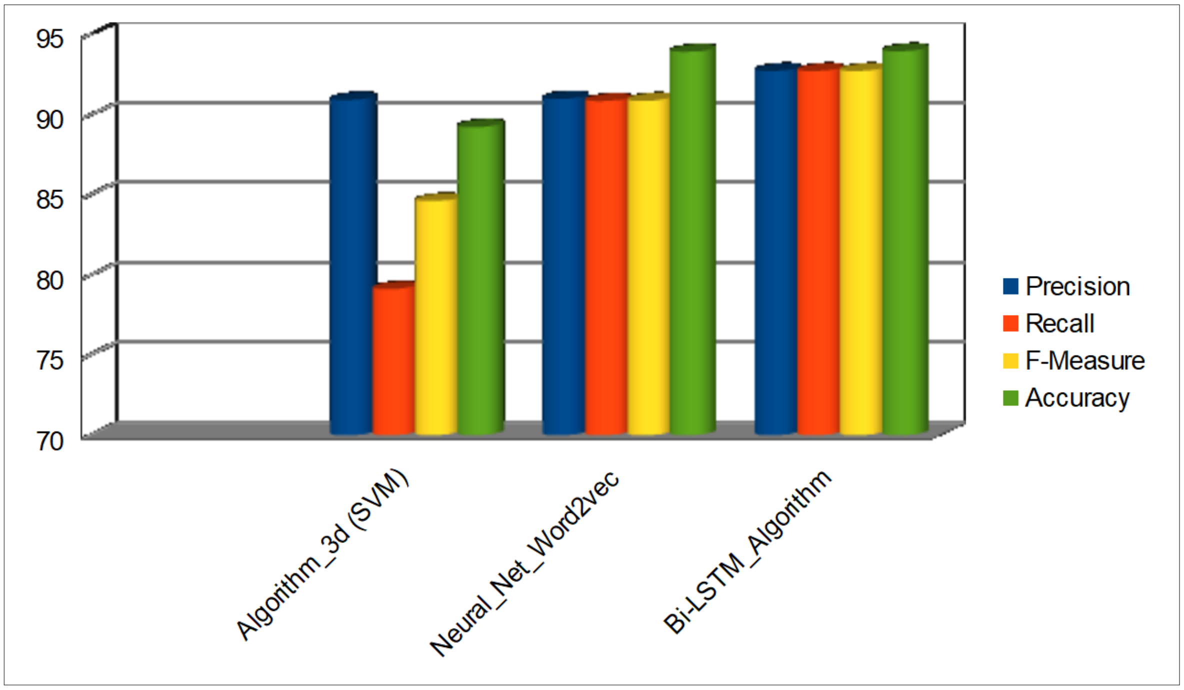
1. Algorithm_3d (Vectorization model with SVM classifier)
2. Neural_Net_Word2vec
3. Bi-LSTM_Algorithm

Table_05 shows the tabulation of performance values of the above three algorithms.

Metrics -->	Precision	Recall	F-Measure	Accuracy
Algorithm_3d (SVM)	90.917 %	79.116 %	84.607 %	89.218 %
Neural_Net_Word2vec	90.875 %	90.843 %	90.858 %	93.917 %
Bi-LSTM_Algorithm	92.715 %	92.715 %	92.715 %	93.945 %

Table_05: Comparing the performance values of the three derived algorithms

The graph below shows the comparison of parameters of the three algorithms. Note that Bi-LSTM_Algorithm gives the best parameter values overall.



We have now narrowed down to three algorithms as seen in table_05. At first, three of the algorithms were built on top of each other and the one with the best values was Algorithm_3d. Two models were implemented using recurrent neural networks. The one with Word2vec embedding implemented was chosen as the one to give netter performance values (Neural_Net_Word2vec). And finally using more complex data processing steps and the concept of bidirectional LSTM layers another model was introduced, Bi-LSTM_Algorithm.

Why Neural_Net_Word2vec gave better results than Algorithm_3d:
Neural_Net_Word2vec is the algorithm based on RNNs and more specifically LSTM layers while Algorithm_3d made use of SVM classifier. It is already noted that among the four classifiers Algorithm_3d was implemented with (KNN, DTC, RF and SVM), SVM gave the best performance values. Neural_Net_Word2vec was based on LSTM network implemented with Word2vec embedding. We can observe that Algorithm_3d gives slightly better precision value when compared to the other algorithm. However, its recall value is much lesser when compared to its counterpart. This is the same case for the accuracy as well. Let us take a look at how precision and recall are measured.

Precision is defined as the value of what propagation of positive identifications were actually right [158]. So if we are to break it down to the formula:
Precision = True_Positives / (True_Positives + False_Positive) --(1)

Recall is defined as the value of what propagation of actual positives were identified correctly [158].

Recall = True_Positive / (True_Positive + False_Negative) – (2)

In the above two formulas:

True Positive is the outcome of the correctly predicted positive class by the model

False Positive is the outcome where the model incorrectly predicts the positive class.

False Negative is an outcome where the model incorrectly predicts the negative class.

If we look into the two formulas, we can see that there should be lesser number of false positives and false negatives in order to be more accurate. The precision and recall values are based on this and the F-measure gives a balance between the precision and recall. If Neural_Net_Word2vec is giving 90.843% of recall value and Algorithm_3d is giving 79.116% recall value, it means that there were a larger number of false negatives derived in the latter and fewer false negatives in the former. A better model or algorithm gives lesser false positives and false negatives. Finally, if we look at the F-measure which is the balance of the precision and recall scores, we can see that Neural_Net_Word2vec gives 90.858% F-score and Algorithm_3d gives 84.607% F-score. So there is a greater balance of precision and recall in the former as compared to the latter. This is because of the following:

* LSTM networks are doing a better job in eliminating the false negatives. This means more correct predictions are made.

* Word2Vec embedding is bringing in the concept of context based analysis. Thanks to this feature, a word is matched to its more probable context which improves the accuracy.

* Neural_Net_Word2vec algorithm is implemented using Keras which provides us with a feature to input the number of training cycles. This is known as epochs. The number of epochs defines the number of times the model is subjected to training. This running of multiple cycles of training can improve the accuracy and precision, a feature that is clearly lacking in the other model.

These are the reasons why Neural_Net_Word2vec is a better model than Algorithm_3d.

Reasons why Bi-LSTM_Algorithm gives better performance values compared to the other two algorithms:

This algorithm made use of functions that pre process the data much better than the previous algorithms did. It was also made domain agnostic by training a number of similar text data sets and storing (pickling in Python) the model. Every time the model is run, it loads it from the model file. This was where Bidirectional LSTM was used in place of traditional LSTM layers. Even here, Keras was made use of. Bidirectional LSTM layers have the capability to retrieve memory over a long period of time and can get data from both the past and the future. This makes the layer more accurate and precise.

So to mention in brief what are the strengths of the final algorithm and why it gives the best results:

* The data pre-processing steps are more complex. Since this algorithm is context based, we cannot remove stop words or tags like in the case of Algorithm_3c where NER-Tagging was used to remove particular nouns.

* The pre-processing steps included creating a list of custom stop words, cleaning rows with very little data, tokenizing sentences and filtering out bi-grams and tri-grams. This made the cleaned data to be processed better.

* Bidirectional LSTM networks have the advantage over traditional LSTM networks that they can not only retain memory over a longer period of time but also can receive input data from both past and future nodes. This way, in Bi-LSTM_Algorithm, each layer or node receives data from both the past layer or node and the future layer or node.

* To improve the speed slightly, Dask was used. Dask provides functionalities to create a data frame using the concept of lazy loading which makes the process of loading a lot faster.

9. Conclusions and Future Work

The methodology describes the flow of how in each step, either a better model was adopted to address the shortcomings of the previous model or a feature was added to the existing model to make it work better and give better performance. The research work first showed the basic concept of sentiment analysis by taking a small example of sentiment analysis using Textblob on an unlabelled data set. Then Twitter sentiment analysis was introduced for which a model was developed which would later be used on the real email data set. The model was used on the email data set but the performance metrics like precision, recall and accuracy were low. In order to better the performance, there was an attempt to make the steps of processing raw text data even better by introducing NER-tagging. Using this and the *dropna()* function which removes null value, a better model was made. The concept of vectorization was studied and a model was implemented to bring out even better values. But this model which made use of vectorization was built on top of the previous one. These models made use of four classifiers out of which SVM gave the best performance values. Although the precision value was quite high, the recall value was relatively lower which meant that the false negatives or the negative values which are falsely classified were more. In order to address this issue, the concept of ANNs was adopted. A model was built using the concepts of RNNs and more specifically LSTM. This model gave both high precision and well as high recall values. A brief research about the concept of Word2vec encouraged me to implement it in the existing model. By doing so, marginally higher values were achived. But considering a good model is the one with not only high performance metrics but also a domain agnostic one, the need to come up with another model was required which made use of the concepts of Bi-LSTM. The model also adopted better steps to process the raw data. Apart from that, a few minor features like using Dask to improve the execution and loading times were helpful.

The results section gave us a clear understanding as to why Bi-LSTM_Algorithm is the best algorithm to address the problem statement. We have answered the research question by providing the best algorithm and the reasons are mentioned. Better and smarter steps to process raw data, making use of lists to store custom stop-words, and the usage of bidirectional LSTM layers definitely gave the model an upper hand. We have compared the different algorithms and seen the drawbacks of each, and we have also seen how the drawbacks are addressed in the next model. Although the model presented is the best among the algorithms presented in the thesis, it does come with some shortcomings. The execution is quite slow. Dask framework is used to address this issue which has made the execution slightly faster. But there still remains the requirement to improve the execution speed. Based on my literature survey, there were papers and articles which mentioned models similar to this analysing data set similar to the one used here gave better values than the ones derived here. This could be due to better processing of data or better cleaning of data.

Based on the research conducted which is documented in the Literature Review section of this thesis, certain knowledge was obtained about existing knowledge in the domain of sentiment analysis on large data sets such as email. Some of the articles and papers reviewed have worked on email sentiment analysis. Some points about how this research work is different is pointed out. One of the strengths of this research work / thesis is that is introduces a domain agnostic approach which enables data across various domains to be accepted. Some other attributes of this thesis which makes it stand out include the introduction of Dask framework which brings in the concept of parallel computation and some advanced pre-processing steps. These are some of the points which advances existing knowledge which is got by researching other articles.

In the future, I plan to implement the PySpark module which is based on parallel computing. The concept is similar to Dask but can perform even better. A hybrid model of bidirectional LSTM combined with Convolution neural networks (CNN) can be considered. Based on my reading, CNN are designed to automatically and adaptively learn spatial hierarchies of features using a back-propagation algorithm [72].

References

[1] - [Xu, J., Zhang, Y., Wu, Y., Wang, J., Dong, X., & Xu, H. (2015). Citation Sentiment Analysis in Clinical Trial Papers. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, *2015*, 1334–1341.]

[2] - [Riloff E, Wiebe J. Proceedings of the 2003 conference on Empirical methods in natural language processing. Association for Computational Linguistics; 2003. Learning extraction patterns for subjective expressions; pp. 105–12.]

[3] - [Hu M, Liu B. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. Seattle, WA, USA: ACM; 2004. Mining and summarizing customer reviews; pp. 168–77.]

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

[5] - [McKinney, W. (2012). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.".]

[6] - [Loria, S. (2018). *textblob Documentation* (pp. 1-73). Technical report.]

[7] - [Loper, E., & Bird, S. (2002). NLTK: the natural language toolkit. *arXiv preprint cs/0205028*.]

[8] - [Perkins, J. (2010). *Python text processing with NLTK 2.0 cookbook*. Packt Publishing Ltd].

[9] - [By Jalaj Thanaki July 2017
Leverage the power of machine learning and deep learning to extract information from text data]

[10] - [Turney, P. D. (2002, July). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 417-424). Association for Computational Linguistics]

[11] - [Almatrafi, O., Parack, S., & Chavan, B. (2015, January). Application of location-based sentiment analysis using Twitter for identifying trends towards Indian general elections 2014. In *Proceedings of the 9th international conference on ubiquitous information management and communication* (p. 41). ACM.]

[12] - [Tweepy Documentation¶. (n.d.). Retrieved from https://tweepy.readthedocs.io/en/latest/.]

[13] - [Zhu, X., & Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, *3*(1), 1-130.]

[14] - [Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report*, *Stanford*, *1*(12), 2009.]

[15] - [B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?Sentiment classification using machine learningtechniques. InProceedings of the Conference onEmpirical Methods in Natural Language Processing(EMNLP), pages 79–86, 2002.]

[16] - [Christensson, Per. "Emoticon Definition." *TechTerms*. Sharpened Productions, 03 November 2014. Web. 09 October 2019. <https://techterms.com/definition/emoticon>]

[17] - KNN Classification using Scikit-learn. (n.d.). Retrieved from https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-lear

[18] - [Weinberger, K. Q., Blitzer, J., & Saul, L. K. (2006). Distance metric learning for large margin nearest neighbor classification. In Advances in neural information processing systems (pp. 1473-1480).]

[19] - [Peterson, L. E. (2009). K-nearest neighbor. Scholarpedia, 4(2), 1883.]

[20] - [Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. E. (2006). Data pre-processing for supervised leaning. International Journal of Computer Science, 1(2), 111-117.]

[21] - [García, S., Luengo, J., & Herrera, F. (2015). *Data pre-processing in data mining* (pp. 59-139). New York: Springer.]

[22] - [Jiang, R., Banchs, R. E., & Li, H. (2016, August). Evaluating and combining name entity recognition systems. In *Proceedings of the Sixth Named Entity Workshop* (pp. 21-27).]

[23] - [Ahmed, I., & Sathyaraj, R. (2015). Named entity recognition by using maximum entropy. *International Journal of Database Theory and Application*, *8*(2), 43-50.]

[24] - [Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, *12*(Oct), 2825-2830.]

[25] - [A survey on Data Mining approaches for Healthcare - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Advantage-and-Disadvantage-of-Various-Classification-Techniques_tbl1_258285203 [accessed 11 Oct, 2019]

[26] - [Gahukar, G. (2019, January 25). Classification Algorithms in Machine Learning... Retrieved from https://medium.com/datadriveninvestor/classification-algorithms-in-machine-learning-85c0ab65ff4.]

[27] - [Nelson, D. (2019, May 11). Overview of Classification Methods in Python with Scikit-Learn. Retrieved from https://stackabuse.com/overview-of-classification-methods-in-python-with-scikit-learn/.]

[28] - [Dreiseitl, S., & Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, *35*(5-6), 352-359.]

[29] - [Quora. (2017, June 19). What Are The Advantages Of Logistic Regression Over Decision Trees? Retrieved from https://www.forbes.com/sites/quora/2017/06/19/what-are-the-advantages-of-logistic-regression-over-decision-trees/#972d1902c352.]

[30] - [Jansen, S. (n.d.). Hands-On Machine Learning for Algorithmic Trading. Retrieved from https://www.oreilly.com/library/view/hands-on-machine-learning/9781789346411/e17de38e-421e-4577-afc3-efdd4e02a468.xhtml.]

[31] - [Zhang, Y., Jin, R., & Zhou, Z. H. (2010). Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, *1*(1-4), 43-52.]

[32] - [Barry, J. (2017). Sentiment Analysis of Online Reviews Using Bag-of-Words and LSTM Approaches. In *AICS* (pp. 272-274).]

[33] - [Brownlee, J. (2019, August 7). A Gentle Introduction to the Bag-of-Words Model. Retrieved from https://machinelearningmastery.com/gentle-introduction-bag-words-model/.]

[34] - [Brownlee, J. (2019, September 3). A Tour of Machine Learning Algorithms. Retrieved from https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/.]

[35] - [*Basheer, I.A., & Hajmeer, M.N. (2000). Artificial neural networks: fundamentals, computing, design, and application. Journal of microbiological methods, 43 1, 3-31 .]*

[36] - [MijwilFollowYüksek, M. M. (n.d.). Artificial Neural Networks Advantages and Disadvantages. Retrieved from https://www.linkedin.com/pulse/artificial-neural-networks-advantages-disadvantages-maad-m-mijwel/.]

[37] - [Dacombe, J. (2017, October 23). An introduction to Artificial Neural Networks (with example). Retrieved from https://medium.com/@jamesdacombe/an-introduction-to-artificial-neural-networks-with-example-ad459bb6941b.]

[38] - [Maind, S. B., & Wankar, P. (2014). Research paper on basic of artificial neural network. *International Journal on Recent and Innovation Trends in Computing and Communication*, *2*(1), 96-100.]

[39] - [Were, K., Bui, D. T., Dick, Ø. B., & Singh, B. R. (2015). A comparative assessment of support vector regression, artificial neural networks, and random forests for predicting and mapping soil organic carbon stocks across an Afromontane landscape. *Ecological Indicators*, *52*, 394-403.]

[40] - [LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.]

[41] - [Deng, L., & Yu, D. (2014). Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, *7*(3–4), 197-387.]

[42] - [What is a Deep Neural Network? - Definition from Techopedia. (n.d.). Retrieved from https://www.techopedia.com/definition/32902/deep-neural-network.]

[43] - [Deep Learning: How Will It Change Healthcare? (2019, January 28). Retrieved from https://orbograph.com/deep-learning-how-will-it-change-healthcare/.]

[44] - [RF Wireless World. (n.d.). Retrieved from https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Deep-Learning.html.]

[45] - [Venkatachalam, M. (2019, June 22). Recurrent Neural Networks. Retrieved from https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce.]

[46] - [A Beginner's Guide to LSTMs and Recurrent Neural Networks. (n.d.). Retrieved from https://skymind.ai/wiki/lstm.]

[47] - [Prajjwal. (2018, May 23). Everything you need to know about Recurrent Neural Networks. Retrieved from https://medium.com/ai-journal/lstm-gru-recurrent-neural-networks-81fe2bcdf1f9.]

[48] - https://pdfs.semanticscholar.org/4b80/89bc9b49f84de43acc2eb8900035f7d492b2.pdf]

[49] - [Gulli, A., & Pal, S. (2017). *Deep Learning with Keras*. Packt Publishing Ltd.]

[50] - [Ratings and Reviews for New Movies and TV Shows. (n.d.). Retrieved from https://www.imdb.com/.]

[51] - [Liang, H., Sun, X., Sun, Y., & Gao, Y. (2017). Text feature extraction based on deep learning: a review. *EURASIP journal on wireless communications and networking*, *2017*(1), 1-12.]

[52] - [[V Singh, B Kumar, T Patnaik, Feature extraction techniques for handwritten text in various scripts: a survey. International Journal of Soft Computing and Engineering 3(1), 238–241 (2013)]]

[53] - [Lai, S., Xu, L., Liu, K., & Zhao, J. (2015, February). Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.]

[54] - [Source code for nltk.tokenize.stanford. (n.d.). Retrieved from https://www.nltk.org/_modules/nltk/tokenize/stanford.html.]

[55] - [Word2vec. (2019, September 11). Retrieved from https://en.wikipedia.org/wiki/Word2vec.]

[56] - [Lewis, D. D., & Ringuette, M. (1994, April). A comparison of two learning algorithms for text categorization. In Third annual symposium on document analysis and information retrieval (Vol. 33, pp. 81-93).]

[57] - [Ali, J., Khan, R., Ahmad, N., & Maqsood, I. (2012). Random forests and decision trees. International Journal of Computer Science Issues (IJCSI), 9(5), 272.].

[58] - [Thanh Noi, P., & Kappas, M. (2018). Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using Sentinel-2 imagery. Sensors, 18(1), 18.]

[60] - [A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://archive.ics.uci.edu/ml]

[61] - [U.S. Geological Survey. Available online: https://earthexplorer.usgs.gov/ (accessed on 22 July 2017).]

[62] - [Liaw, A.; Wiener, M. Classification and regression by randomForest. R News 2002, 2, 18–22.]

[63] - [Breiman, L. Random forests. Mach. Learn. 2001, 45, 5–32.]

[64] - [Duda, R.; Hart, P. Pattern Classification and Scene Analysis; John Wiley & Sons: New York, NY, USA, 1973.]

[65] - [Nowak, J., Taspinar, A., & Scherer, R. (2017, June). LSTM recurrent neural networks for short text and sentiment classification. In International Conference on Artificial Intelligence and Soft Computing (pp. 553-562). Springer, Cham.]

[66] - [Graves, A., Fernández, S., & Schmidhuber, J. (2005, September). Bidirectional LSTM networks for improved phoneme classification and recognition. In International Conference on Artificial Neural Networks (pp. 799-804). Springer, Berlin, Heidelberg.]

[67] - [Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11), 2673-2681.]

[68] - [J.S.Garofolo,L.F.Lamel,W.M.Fisher,J.G.Fiscus,D.S.Pallett,,andN.L.Dahlgren.Darpa timit acoustic phonetic continuous speech corpus cdrom, 1993]

[69] - [Shuang Bi, W.Z.: Cs294-1 final project algorithms comparison deep learningneural network — adaboost — random forest.http://bid.berkeley.edu/cs294-1-spring13/images/0/0d/ProjectReport(Shuang_and_Wenchang).pdf(May 15, 2013)]

[70] - [Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectionallstm and other neural network architectures. Neural Networks18(5) (2005) 602–610]

[71] - [J.S.Garofolo,L.F.Lamel,W.M.Fisher,J.G.Fiscus,D.S.Pallett,,andN.L.Dahlgren.Darpa timit acoustic phonetic continuous speech corpus cdrom, 1993.]

[72] - [Yamashita, R., Nishio, M., Gian, R. K., & Togashi, K. (2018, June 22). Convolutional neural networks: an overview and application in radiology. Retrieved from https://link.springer.com/article/10.1007/s13244-018-0639-9.]

[73] - [Li, H. (2017). Deep learning for natural language processing: advantages and challenges. National Science Review.]

[74] - [Schroff F, Kalenichenko D and Philbin J.CVPR2015.]

[75] - [Yang Y, Sun J and Li Het al. NIPS2016]

[76] - [Wang, J., Yu, L. C., Lai, K. R., & Zhang, X. (2016, August). Dimensional sentiment analysis using a regional CNN-LSTM model. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 225-230).]

[81] - [Vanzo, A., Croce, D., & Basili, R. (2014, August). A context-based model for sentiment analysis in twitter. In Proceedings of coling 2014, the 25th international conference on computational linguistics: Technical papers (pp. 2345-2354).]

[82] - [Geva. (n.d.). Using the Keras Flatten Operation in CNN Models with Code Examples. Retrieved from https://missinglink.ai/guides/keras/using-keras-flatten-operation-cnn-models-code-examples/.]

[83] - [DataFrame¶. (n.d.). Retrieved from https://docs.dask.org/en/latest/dataframe.html.]

[84] - [Dask Executor¶. (n.d.). Retrieved from https://airflow.apache.org/howto/executor/use-dask.html.]

[85] - [Rocklin, M. (n.d.). GPU Dask Arrays, first steps throwing Dask and CuPy together. Retrieved from http://matthewrocklin.com/blog/work/2019/01/03/dask-array-gpus-first-steps.]

[86] - Meena, S. (2019, February 8). Your Guide to Sentiment Analysis. Retrieved from https://medium.com/seek-blog/your-guide-to-sentiment-analysis-344d43d225a7.

[87] - [Sentiment Analysis. (n.d.). Retrieved from https://www.sciencedirect.com/topics/engineering/sentiment-analysis.]

[88] - [Vohra, S. M., & Teraiya, J. B. (2013). A comparative study of sentiment analysis techniques. *Journal JIKRCE*, *2*(2), 313-317.]

[89] - [Gupta, S. (2018, March 5). Applications of Sentiment Analysis in Business. Retrieved from https://itnext.io/applications-of-sentiment-analysis-in-business-4267e3c622fc.]

[90] - [Boot, A. B., Sang, E. T. K., Dijkstra, K., & Zwaan, R. A. (2019, July 9). How character limit affects language usage in tweets. Retrieved from https://www.nature.com/articles/s41599-019-0280-3.]

[91] - [Choosing the Right Machine Learning Algorithm. (n.d.). Retrieved from https://hackernoon.com/choosing-the-right-machine-learning-algorithm-68126944ce1f.]

[92] - [Biological Neural Network. (n.d.). Retrieved from https://www.sciencedirect.com/topics/veterinary-science-and-veterinary-medicine/biological-neural-network.]

[93] - [Ashely, & Vinita. (1970, August 3). Advantages and disadvantages of neural networks. Retrieved from https://intellipaat.com/community/21886/advantages-and-disadvantages-of-neural-networks.]

[94] - [Mahapatra, S. (2019, January 22). Why Deep Learning over Traditional Machine Learning? Retrieved from https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063.]

[95] - [RF Wireless World. (n.d.). Retrieved from https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Deep-Learning.html.]

[96] - [Mittal, A. (2019, October 12). Understanding RNN and LSTM. Retrieved from https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e.]

[97] - [Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, *5*(1), 1-167.]

[98] - [Collomb, A., Costea, C., Joyeux, D., Hasan, O., & Brunie, L. (2014). A study and comparison of sentiment analysis methods for reputation evaluation. *Rapport de recherche RR-LIRIS-2014-002*.]

[99] - [5 Things You Need to Know about Sentiment Analysis and Classification. (n.d.). Retrieved from https://www.kdnuggets.com/2018/03/5-things-sentiment-analysis-classification.html.]

[100] - [Pandey, Avinash & Rajpoot, Dharmveer & Saraswat, Mukesh. (2017). Twitter sentiment analysis using hybrid cuckoo search method. Information Processing & Management. 53. 764-779. 10.1016/j.ipm.2017.02.004.]

[101] - [Garbade, M. J. (2018, October 15). A Simple Introduction to Natural Language Processing. Retrieved from https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32.]

[102] - [Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, *12*(Aug), 2493-2537.]

[103] - [Annual Review of Information Science and Technology ISSN ... (n.d.). Retrieved from https://strathprints.strath.ac.uk/2611/1/strathprints002611.pdf.]

[104] - [9 Examples of Natural Language Processing. (n.d.). Retrieved from https://simplicable.com/new/natural-language-processing.]

[105] - [Saslow, E. (2018, November 8). Unsupervised Machine Learning. Retrieved from https://towardsdatascience.com/unsupervised-machine-learning-9329c97d6d9f.]

[106] - [Rouse, M., Haughn, M., Rouse, M., & Rouse, M. (n.d.). What is unsupervised learning? - Definition from WhatIs.com. Retrieved from https://whatis.techtarget.com/definition/unsupervised-learning.]

[107] - [Brownlee, J. (2019, August 12). Supervised and Unsupervised Machine Learning Algorithms. Retrieved from https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/.]

[108] - [Unsupervised Machine Learning: What is, Algorithms, Example. (n.d.). Retrieved from https://www.guru99.com/unsupervised-machine-learning.html#2.]

[109] - [Supervised Learning. (n.d.). Retrieved from https://www.sciencedirect.com/topics/computer-science/supervised-learning.]

[110] - [Pros and Cons of Supervised Machine Learning. (2019, October 10). Retrieved from https://pythonistaplanet.com/pros-and-cons-of-supervised-machine-learning/.]

[111] - [Osiński, B. (2019, July 5). What is reinforcement learning? The complete guide. Retrieved from https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/.]

[112] - [Sanders, J. (2019, October 18). Python programming language: A cheat sheet. Retrieved from https://www.techrepublic.com/article/python-programming-language-a-cheat-sheet/.]

[113] - [NumPy¶. (n.d.). Retrieved from https://numpy.org/.]

[114] - [numpy.mean¶. (n.d.). Retrieved from https://docs.scipy.org/doc/numpy/reference/generated/numpy.mean.html.]

[115] - [numpy.mean¶. (n.d.). Retrieved from https://docs.scipy.org/doc/numpy/reference/generated/numpy.mean.html.]

[116] - S[numpy.zeros¶. (n.d.). Retrieved from https://docs.scipy.org/doc/numpy/reference/generated/numpy.zeros.html.]

[117] - [numpy.sum¶. (n.d.). Retrieved from https://docs.scipy.org/doc/numpy/reference/generated/numpy.sum.html.]

[118] - [Pandas Basics. (n.d.). Retrieved from https://www.learnpython.org/en/Pandas_Basics.]

[119] - [re - Regular expression operations¶. (n.d.). Retrieved from https://docs.python.org/3/library/re.html.]

[120] - [6.1. string - Common string operations¶. (n.d.). Retrieved from https://docs.python.org/3.6/library/string.html.]

[121] - [Satyabrata PalI am a software engineer based out of Pune. (2019, July 5). Scikit-learn Tutorial: Machine Learning in Python. Retrieved from https://www.dataquest.io/blog/sci-kit-learn-tutorial/.]

[122] - [texttables python module¶. (n.d.). Retrieved from https://texttables.readthedocs.io/en/latest/.]

[123] - [What is Tokenization? - Definition from Techopedia. (n.d.). Retrieved from https://www.techopedia.com/definition/13698/tokenization.]

[124] - [Code Faster with Line-of-Code Completions, Cloudless Processing. (n.d.). Retrieved from https://kite.com/python/docs/nltk.word_tokenize.]

[125] - [Tokenizing Words and Sentences with NLTK. (2016, August 26). Retrieved from https://pythonspot.com/tokenizing-words-and-sentences-with-nltk/.]

[126] - [NLTK stop words. (2016, August 26). Retrieved from https://pythonspot.com/nltk-stop-words/.]

[127] - [Li, S. (2018, December 6). Named Entity Recognition with NLTK and SpaCy. Retrieved from https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da.]

[128] - [Simplified Text Processing¶. (n.d.). Retrieved from https://textblob.readthedocs.io/en/dev/.]

[129] - [Aggarwal, S. (1969, August 1). Sentiment Analysis with TextBlob and Python. Retrieved from https://linuxhint.com/sentiment_analysis_textblob_python/.]

[130] - [(2015, June 7). Retrieved from https://planspace.org/20150607-textblob_sentiment/.]

[131] - [Nausheen, F., & Begum, S. H. (2018, January). Sentiment analysis to predict election results using Python. In *2018 2nd International Conference on Inventive Systems and Control (ICISC)* (pp. 1259-1262). IEEE.]

[132] - [Vijayarani, S., & Janani, R. (2016). Text mining: open source tokenization tools-an analysis. *Advanced Computational Intelligence: An International Journal (ACII)*, *3*(1), 37-47.]

[133] - [Seif, G. (2019, May 4). An Introduction to Scikit Learn: The Gold Standard of Python Machine Learning. Retrieved from https://towardsdatascience.com/an-introduction-to-scikit-learn-the-gold-standard-of-python-machine-learning-e2b9238a98ab.]

[134] - [Xue, B. (n.d.). Hardware and Software for Machine LearningScikit-learn]

[135] - [sklearn.model_selection.train_test_split¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.]

[136] - [API Reference¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics.]

[137] – [https://www.researchgate.net/publication/328103122_An_approach_to_analyse_suicidal_tendency_in_blogs_and_tweets_using_Sentiment_Analysis]

[138] - [Harrison, O. (2019, July 14). Machine Learning Basics with the K-Nearest Neighbors Algorithm. Retrieved from https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.]

[139] - [Srivastava, T., & Srivastava, T. (2019, September 3). Introduction to KNN, K-Nearest Neighbors : Simplified. Retrieved from https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/.]

[140] - [Schott, M. (2019, April 26). K-Nearest Neighbors (KNN) Algorithm for Machine Learning. Retrieved from https://medium.com/capital-one-tech/k-nearest-neighbors-knn-algorithm-for-machine-learning-e883219c8f26.]

[141] - [Bronshtein, A. (2019, May 6). A Quick Introduction to K-Nearest Neighbors Algorithm. Retrieved from https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7.]

[142] - [Genesis. (2018, September 25). Pros and Cons of K-Nearest Neighbors. Retrieved from https://www.fromthegenesis.com/pros-and-cons-of-k-nearest-neighbors/.]

[143] - [Quantitative Research: Definition, Methods, Types and Examples. (2019, September 6). Retrieved from https://www.questionpro.com/blog/quantitative-research/.]

[144] - [Introduction to Machine Learning Algorithms: Logistic Regression. (n.d.). Retrieved from https://hackernoon.com/introduction-to-machine-learning-algorithms-logistic-regression-cbdd82d81a36.]

[145] - [[Introduction to Machine Learning Algorithms: Logistic Regression. (n.d.). Retrieved from https://hackernoon.com/introduction-to-machine-learning-algorithms-logistic-regression-cbdd82d81a36.]

[146] - [The Logistic Regression Algorithm. (2019, January 8). Retrieved from https://www.experfy.com/blog/the-logistic-regression-algorithm.]

[147] - [Molnar, C. (2019, September 18). Interpretable Machine Learning. Retrieved from https://christophm.github.io/interpretable-ml-book/logistic.html.]

[148] - [Admin. (2019, September 1). The Logistic Regression Algorithm-Detailed Overview. Retrieved from https://kraj3.com.np/blog/2019/08/the-logistic-regression-algorithm-detailed-overview/.]

[149] - [How the random forest algorithm works in machine learning. (2017, October 1). Retrieved from https://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/.]

[150] - [Synced, Dennis, Machine Intelligence | Technology & Industry | Information & Analysis, & Machine Intelligence | Technology & Industry | Information & Analysis. (2017, October 24). How Random Forest Algorithm Works in Machine Learning. Retrieved from https://syncedreview.com/2017/10/24/how-random-forest-algorithm-works-in-machine-learning/.]

[151] - [(n.d.). Retrieved from https://www.saedsayad.com/decision_tree_overfitting.htm.]

[152] - [Kumar, N. (n.d.). Advantages and Disadvantages of Random Forest Algorithm in Machine Learning. Retrieved from http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html.]

[153] - [Www.statsoft.com. (n.d.). Retrieved from http://www.statsoft.com/textbook/support-vector-machines.]

[154] - [Classifying data using Support Vector Machines(SVMs) in Python. (2017, May 23). Retrieved from https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-python/.]

[155] - [Tang, J., Li, H., Cao, Y., & Tang, Z. (2005, August). Email data cleaning. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (pp. 489-498). ACM.]

[156] - [Kumar, D. (2019, June 13). Top 4 advantages and disadvantages of Support Vector Machine or SVM. Retrieved from https://medium.com/@dhiraj8899/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107.]

[157] - [Classification: True vs. False and Positive vs. Negative. (n.d.). Retrieved from https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative.]

[158] - [Classification: Precision and Recall | Machine Learning Crash Course. (n.d.). Retrieved from https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall.]

[159] - [Shung, K. P. (2018, June 8). Accuracy, Precision, Recall or F1? Retrieved from https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.]

[160] - [Recurrent Neural Networks cheatsheet Star. (n.d.). Retrieved from https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.]

[161] - [Bidirectional Recurrent Neural Networks. (2019, May 17). Retrieved from https://deepai.org/machine-learning-glossary-and-terms/bidirectional-recurrent-neural-networks.]

[162] - [Lee, C. (2018, March 5). Understanding Bidirectional RNN in PyTorch. Retrieved from https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66.]

[163] - [Keras: The Python Deep Learning library. (n.d.). Retrieved from https://keras.io/.]

[164] - [Tanner, G. (2019, March 6). Introduction to Deep Learning with Keras. Retrieved from https://towardsdatascience.com/introduction-to-deep-learning-with-keras-17c09e4f0eb2.]

[165] - [TensorFlow. (n.d.). Retrieved from https://www.tensorflow.org/.]

[166] - [(n.d.). Retrieved from https://cran.rstudio.com/web/packages/keras/vignettes/about_keras_layers.html.]

[168] - [Wang, W., & Chang, B. (2016). Graph-based dependency parsing with bidirectional LSTM. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 2306-2315).]

[169] - [scikit-learn Review: Pricing, Pros, Cons & Features. (2019, June 9). Retrieved from http://comparecamp.com/scikit-learn-review-pricing-pros-cons-features/.]

[170] - [Britz, D. (2016, July 8). Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs. Retrieved from http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.]

[171] - [Rouse, M., Laskowski, N., Rouse, M., & Rouse, M. (n.d.). What is recurrent neural networks? - Definition from WhatIs.com. Retrieved from https://searchenterpriseai.techtarget.com/definition/recurrent-neural-networks.]

[172] - [Abraham, A. (2005). Artificial neural networks. Handbook of measuring system design.]

[173] - [Named Entity Recognition: A Practitioner's Guide to NLP. (n.d.). Retrieved from https://www.kdnuggets.com/2018/08/named-entity-recognition-practitioners-guide-nlp-4.html.]

[174] - [Yildirim, Ö. (2018, March 28). A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification. Retrieved from https://www.sciencedirect.com/science/article/pii/S0010482518300738.]

[175] - [Dask¶. (n.d.). Retrieved from https://docs.dask.org/en/latest/.]

[176] - [Bakharia, A. (2018, June 6). Why every Data Scientist should use Dask? Retrieved from https://towardsdatascience.com/why-every-data-scientist-should-use-dask-81b2b850e15b.]

[177] - [Alhonte, M. (2019, August 10). Lazy Pandas and Dask. Retrieved from https://hackersandslackers.com/cutting-a-file-down-to-size-with-dask.]

[178] - [Grover, P. (2019, May 10). Speeding up your Algorithms Part 4- Dask. Retrieved from https://towardsdatascience.com/speeding-up-your-algorithms-part-4-dask-7c6ed79994ef.]

[179] - [Banerjee, S. (2018, October 21). Word2Vec - a baby step in Deep Learning but a giant leap towards Natural Language Processing. Retrieved from https://medium.com/explore-artificial-intelligence/word2vec-a-baby-step-in-deep-learning-but-a-giant-leap-towards-natural-language-processing-40fe4e8602ba.]

[180] - [(n.d.). Retrieved from https://lh3.googleusercontent.com/--jJMbpj-6YI/WP2ZA7QGCTI/AAAAAAAAAfg/_hUadNcxZ6MKCDCdJaagKz6Luon3adLWwCHM/s1600-h/sa%5B5%5D.]

Appendix A

This part of the appendix shows the psuedocodes for the algorithms.

1. **Algorithm_1a**

```
""" IMPORTING PYTHON MODULES """
def main():
    dframe = pandas.read_csv ("/home/san1234/thesis/write-up/coursera_small.csv")
    del dframe ['Label']

    ls_label = []
    ls_rev = list (dframe['Review'])

    positive_score = 0
    neutral_score  = 0
    negative_score = 0

    i = 0
    start = time.time()
    while (i < len(ls_rev)):
        blob = TextBlob (ls_rev[i])
        polarity = blob.sentiment.polarity

        if (polarity > 0.5):
            ls_label.append (5)
            positive_score += 1
        elif (polarity > 0.0 and polarity <= 0.5):
            ls_label.append (4)
            positive_score += 1
        elif (polarity == 0.0):
            ls_label.append (3)
            neutral_score += 1
        elif (polarity > -0.5 and polarity < 0.0):
            ls_label.append (2)
            negative_score += 1
        elif (polarity <= -0.5):
            ls_label.append (1)
            negative_score += 1
        else:
            ls_label.append ("INVALID")

        i += 1
    stop = time.time()

    dframe ['Labels'] = ls_label
    print ("\n\n")

    if (positive_score > negative_score and positive_score > neutral_score):
        print ("\n\nThe data has POSITIVE sentiment.")
    elif (negative_score > positive_score and negative_score > neutral_score):
        print ("\n\nThe dataset has NEGATIVE sentiment.")
    elif (neutral_score > positive_score and neutral_score > negative_score):
        print ("\n\nThe dataset has NEUTRAL sentiment.")



    total_time = (stop - start)
    print ("\n\nTotal time --> {} ms\n".format (total_time * 1000))

main()
```

2. **Algorithm_1b**

```
""" IMPORTING REQUIRES PYTHON MODEULES """
def process():
    df = pandas.read_csv ("/home/san1234/thesis/write-up/coursera_small.csv", encoding = "ISO-
8859-1")
    dframe = pandas.DataFrame()
    dframe ['Review'] = df ['Review']
    dframe ['Labels']  = df ['Label']

    fin_ls = []

    for i in dframe ['Review']:
        str1 = pre(i)
        fin_ls.append(str1)
    del dframe ['Review']
    dframe ['Reviews'] = fin_ls
    success_token = "SUCCESS..."
    return (dframe, success_token)

def pre(example):
    word = nltk.word_tokenize (example)

    stop_words = list (stopwords.words ("english"))
    filtered_sent = []
    for i in word:
        if i not in stop_words:
            filtered_sent.append  (i)
    str1 = ''
    for i in filtered_sent:
        str1 = str1 + i + ' '

    example = str1
    ps = PorterStemmer()
    word = nltk.word_tokenize (example)
    ls = []
    for w in word:
        ls.append (ps.stem (w))
    str1 = ''
    for i in ls:
        str1 = str1 + i + ' '

    example = str1

    word = nltk.word_tokenize (example)
    word = [word.lower() for word in word if word.isalpha()]

    str1 = ''
    for i in word:
        str1 = str1 + i + ' '

    return (str1)

import pandas
def main():
    dframe, success_token = process()
    del dframe ['Labels']

    ls_label = []
    ls_rev = list (dframe['Reviews'])

    positive_score = 0
    neutral_score  = 0
    negative_score = 0

    i = 0
```

```
start = time.time()
while (i < len(ls_rev)):
    blob = TextBlob (ls_rev[i])
    polarity = blob.sentiment.polarity

    if (polarity > 0.5):
        ls_label.append (5)
        positive_score += 1
    elif (polarity > 0.0 and polarity <= 0.5):
        ls_label.append (4)
        positive_score += 1
    elif (polarity == 0.0):
        ls_label.append (3)
        neutral_score += 1
    elif (polarity > -0.5 and polarity < 0.0):
        ls_label.append (2)
        negative_score += 1
    elif (polarity <= -0.5):
        ls_label.append (1)
        negative_score += 1
    else:
        ls_label.append ("INVALID")

    i += 1
stop = time.time()

dframe ['Labels'] = ls_label
print ("\n\n")

if (positive_score > negative_score and positive_score > neutral_score):
    print ("\nThe data has POSITIVE sentiment.")
elif (negative_score > positive_score and negative_score > neutral_score):
    print ("\n\nThe dataset has NEGATIVE sentiment.")
elif (neutral_score > positive_score and neutral_score > negative_score):
        print ("\n\nThe dataset has NEUTRAL sentiment.")



    total_time = (stop - start)
    print ("\n\nTotal time --> {} ms".format (total_time * 1000))

main()
```

3. **Algorithm_2a**

```
""" IMPORTING REQUIRED PYTHON MODULES """

def percentage (part, whole):
    return (100 * float (part) / float (whole))

consumerKey = "zB0rMPOldUVpmQnMQFGAIG9nh"
consumerSecret = "IevXrHjv8qfdB7DkuBda9sXTeG8TAapffZFvyv0Sn62vR3ZJvb"
accessToken = "1170896970400927744-Qj9IaO6SghaaEjGs56nojLUOye8yhq"
accessTokenSecret = "eJ9rroWpQIRmBiaTVO6ADdLV50R6Lha9QIUrwpimMBHBj"

auth = tweepy.OAuthHandler (consumerKey, consumerSecret)
auth.set_access_token (accessToken, accessTokenSecret)
api = tweepy.API (auth)

searchTerm = input ("Enter keyword/hashtag to search about: ")
noOfSearchTerms = int (input ("Enter how many tweets to analyse: "))

tweets = tweepy.Cursor (api.search, q = searchTerm, language = "English").items
(noOfSearchTerms)

positive = 0
neutral = 0
negative = 0

polarity = 0                #-- Average of all tweets

for tweet in tweets:
    print (tweet.text)
    analysis = TextBlob (tweet.text)
    polarity += analysis.sentiment.polarity
    if (analysis.sentiment.polarity == 0):
        neutral += 1

    elif (analysis.sentiment.polarity < 0):
        negative += 1

    elif (analysis.sentiment.polarity > 0):
        positive += 1

dc = {"Positive" : positive, "Neutral" : neutral, "Negative" : negative}
print ("\n")
print ("DC --> {}".format (dc))
print ("\n")

positive = percentage (positive, noOfSearchTerms)
negative = percentage (negative, noOfSearchTerms)
neutral  = percentage (neutral,  noOfSearchTerms)
polarity = percentage (polarity, noOfSearchTerms)

positive = format (positive, '.2f')
negative = format (negative, '.2f')
neutral  = format (neutral,  '.2f')

print ("How are people reactine about " + searchTerm + " by analysing " + str (noOfSearchTerms)
+ " Tweets.")

if (polarity == 0):
    print ("Neutral")
elif (polarity < 0):
    print ("Negative")
elif (polarity > 0):
    print ("Positive")
```

4. **Algorithm_2b**

```
""" IMPORTING REQUIRED PYTHON MODULES """

def process():
    df = pandas.read_csv ("GOP_Tweets.csv")
    dframe = pandas.DataFrame()
    dframe ['Review'] = df ['Review']
    dframe ['Labels']  = df ['Label']

    fin_ls = []

    for i in dframe ['Review']:
        str1 = pre(i)
        fin_ls.append(str1)
    del dframe ['Review']
    dframe ['Reviews'] = fin_ls
    return (dframe)

def pre(example):
    word = nltk.word_tokenize (example)

    stop_words = list (stopwords.words ("english"))
    filtered_sent = []
    for i in word:
        if i not in stop_words:
            filtered_sent.append  (i)
    str1 = ''
    for i in filtered_sent:
        str1 = str1 + i + ' '

    example = str1
    ps = PorterStemmer()
    word = nltk.word_tokenize (example)
    ls = []
    for w in word:
        ls.append (ps.stem (w))
    str1 = ''
    for i in ls:
        str1 = str1 + i + ' '

    example = str1

    word = nltk.word_tokenize (example)
    word = [word.lower() for word in word if word.isalpha()]

    str1 = ''
    for i in word:
        str1 = str1 + i + ' '

    return (str1)

def tweet_analysis():
    dframe = process()
    le = pre-processing.LabelEncoder()
    dframe ["Reviews"] = le.fit_transform (dframe ["Reviews"])

    X = dframe ["Reviews"]
    Y = dframe ["Labels"]

    x_train, x_test, y_train, y_test = train_test_split (X, Y, test_size = 0.2, random_state = 1)

    x_train = x_train.values.reshape(-1, 1)
    y_train = y_train.values.reshape(-1, 1)
    x_test = x_test.values.reshape(-1, 1)
    y_test = y_test.values.reshape(-1, 1)
    knn = KNeighborsClassifier (n_neighbors = 3)
    knn.fit (x_train, y_train)
    score = knn.score (x_train, y_train)
    print ("\n\nAccuracy --> {:.3f} %\n".format (score * 100))

tweet_analysis()
```

5. **<u>Algorithm_3a</u>**

```
""" IMPORTING REQUIRED PYTHON MODULES """

def process():
    df = pandas.read_csv ("/home/san1234/thesis/write-up/email_data_200.csv")

    dframe = pandas.DataFrame()
    dframe ['Review'] = df ['Review']
    dframe ['Labels']  = df ['Label']

    """ Each row of email text data is sent to the function pre_process() which cleans the data """

def pre(example):
    word = nltk.word_tokenize (example)

    """ Step-1: Dates and formalities are removed """

    """ Step-2: Stop Words are removed """

    """ Step-3: Words are stemmed and lemmatized """

    """ Step-4: Punctuations are removed from the text """

from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

def tweet_analysis():
    dframe = process()
    le = pre-processing.LabelEncoder()
    dframe ["Reviews"] = le.fit_transform (dframe ["Reviews"])
    X = dframe ["Reviews"]
    Y = dframe ["Labels"]

    x_train, x_test, y_train, y_test = train_test_split (X, Y, test_size = 0.2, random_state = 1)

    x_train = x_train.values.reshape(-1, 1)
    y_train = y_train.values.reshape(-1, 1)
    x_test = x_test.values.reshape(-1, 1)
    y_test = y_test.values.reshape(-1, 1)

    knn = KNeighborsClassifier (n_neighbors = 3)
    knn.fit (x_train, y_train)
    score = knn.score (x_train, y_train)

    y_pred = knn.predict (x_test)
    y_score = knn.score (x_test, y_test)

    prec = precision_score (y_pred, y_test, average = 'weighted')
    recall = recall_score (y_pred, y_test, average = 'weighted')
    fscore = f1_score (y_pred, y_test, average = 'weighted')

    print ("\n\nAccuracy --> {:.3f} %\n".format (score * 100))
    print ("\nPrecision --> {:.3f} %\n".format (prec * 100))
    print ("\nRecall --> {:.3f} %\n".format (recall * 100))
    print ("\nF1_Score --> {:.3f} %\n".format (fscore * 100))

tweet_analysis()
```

6. **<u>Algorithm_3b</u>**

```
""" IMPORTING REQUIRED PYTHON MODULES """

def process():
    df = pandas.read_csv ("/home/san1234/thesis/write-up/email_data_200.csv")

    dframe = pandas.DataFrame()
    dframe ['Review'] = df ['Review']
    dframe ['Labels']  = df ['Label']

    """ Each row of email text data is sent to the function pre_process() which cleans the data """

    """ Once each row of text data is processed, the dropna() function of pandas is used to remove
missing values and null vales """

def pre(example):
    word = nltk.word_tokenize (example)

    """ Step-1: Dates and formalities are removed """

    """ Step-2: Stop Words are removed """

    """ Step-3: NER-Tagging is used to identify names of organizations and proper nouns. Even the
names and signatures are eliminated """

    """ Step-4: Words are stemmed and lemmatized """

    """ Step-5: Punctuations are removed from the text """



def tweet_analysis():
    dframe = process()
    le = pre-processing.LabelEncoder()
    dframe ["Reviews"] = le.fit_transform (dframe ["Reviews"])

    X = dframe ["Reviews"]
    Y = dframe ["Labels"]

    x_train, x_test, y_train, y_test = train_test_split (X, Y, test_size = 0.2, random_state = 1)

    x_train = x_train.values.reshape(-1, 1)
    y_train = y_train.values.reshape(-1, 1)
    x_test = x_test.values.reshape(-1, 1)
    y_test = y_test.values.reshape(-1, 1)

    knn = KNeighborsClassifier (n_neighbors = 3)
    knn.fit (x_train, y_train)
    score = knn.score (x_train, y_train)

    y_pred = knn.predict (x_test)
    y_score = knn.score (x_test, y_test)

    prec = precision_score (y_pred, y_test, average = 'weighted')
    recall = recall_score (y_pred, y_test, average = 'weighted')
    fscore = f1_score (y_pred, y_test, average = 'weighted')

    print ("\n\nAccuracy --> {:.3f} %\n".format (score * 100))
    print ("\nPrecision --> {:.3f} %\n".format (prec * 100))
    print ("\nRecall --> {:.3f} %\n".format (recall * 100))
    print ("\nF1_Score --> {:.3f} %\n".format (fscore * 100))

tweet_analysis()
```

7. **Algorithm_3c**

```
""" IMPORTING REQUIRED PYTHON MODULES """

def process():
    df = pandas.read_csv ("/home/san1234/thesis/write-up/email_data_200.csv")

    dframe = pandas.DataFrame()
    dframe ['Review'] = df ['Review']
    dframe ['Labels']  = df ['Label']

    """ Each row of email text data is sent to the function pre_process() which cleans the data """

    """ Once each row of text data is processed, the dropna() function of pandas is used to remove
missing values and null vales """

def pre(example):
    word = nltk.word_tokenize (example)

    """ Step-1: Dates and formalities are removed """

    """ Step-2: Stop Words are removed """

    """ Step-3: NER-Tagging is used to identify names of organizations and proper nouns. Even the
names and signatures are eliminated """

    """ Step-4: Words are stemmed and lemmatized """

    """ Step-5: Punctuations are removed from the text """
```

```
def main():
    dframe = process()
    col_names = list (dframe.columns)

    label_encoder = pre-processing.LabelEncoder()
    dframe ['Reviews'] = label_encoder.fit_transform(dframe['Reviews'])

    X = dframe ["Reviews"]
    Y = dframe ['Labels']

    x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=1)

    dtc = DecisionTreeClassifier()

    x_train = x_train.values.reshape(-1, 1)
    y_train = y_train.values.reshape(-1, 1)
    x_test = x_test.values.reshape(-1, 1)
    y_test = y_test.values.reshape(-1, 1)


    ''' Implementing the K-Nearest-Neighbors (KNN) classifier and calculating the performance
metrics '''

    ''' Implementing the Decision Tree (DT) classifier and calculating the performance metrics '''

    ''' Implementing the Random Forest (RF) classifier and calculating the performance metrics '''

    ''' Implementing the Support Vector Machine (SVM) classifier and calculating the performance
metrics '''
main()
```

8. **Algorithm_3d**

```
""" IMPORTING REQUIRED PYTHON MODULES """

def process():
    df = pandas.read_csv ("/home/san1234/thesis/write-up/email_data_200.csv")

    dframe = pandas.DataFrame()
    dframe ['Review'] = df ['Review']
    dframe ['Labels']  = df ['Label']

    """ Each row of email text data is sent to the function pre_process() which cleans the data """

    """ Once each row of text data is processed, the dropna() function of pandas is used to remove
missing values and null vales """

def pre(example):
    word = nltk.word_tokenize (example)

    """ Step-1: Dates and formalities are removed """

    """ Step-2: Stop Words are removed """

    """ Step-3: NER-Tagging is used to identify names of organizations and proper nouns. Even the
names and signatures are eliminated """

    """ Step-4: Words are stemmed and lemmatized """

    """ Step-5: Punctuations are removed from the text """

def precision():

    dframe = process()
    dframe.head()

    X = dframe ["Reviews"]
    Y = dframe ["Labels"]

    vectorizer = CountVectorizer()

    x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=1)

    x_train = vectorizer.fit_transform (x_train)
    x_test = vectorizer.fit_transform (x_test)

    feature_names = vectorizer.get_feature_names()

    """ Other important steps of vectorizing """

     ''' Implementing the K-Nearest-Neighbors (KNN) classifier and calculating the performance
metrics '''

    ''' Implementing the Decision Tree (DT) classifier and calculating the performance metrics '''

    ''' Implementing the Random Forest (RF) classifier and calculating the performance metrics '''

    ''' Implementing the Support Vector Machine (SVM) classifier and calculating the performance
metrics '''

precision()
```

9. **Neural_Net_01**

```
""" IMPORTING REQUIRED PYTHON MODULES """

def preprocess():
    df = pd.read_csv('Coursera_Reviews.csv')
    dframe = pd.DataFrame()
    dframe['Review'] = df['Review']
    dframe['Label'] = df['Label']

    print (dframe.columns)

    df_reviews = (dframe ['Review'])
    df_labels  = (dframe ['Label'])
    main_dframe = pd.DataFrame ({'Reviews' : ls_reviews, 'Labels' : ls_labels})
    print (main_dframe)

    model_name = "/home/san1234/thesis/sampling.model"
    model.save(model_name)
    num_features = 300

    top_words = w_model.wv.syn0.shape[0]
    mxlen = 50
    nb_classes = 3

    train_set, test_set = train_test_split (main_dframe, test_size=0.2)

    batch_size = 128
    nb_epoch = 1
    n_timesteps = 10

    embedding_layer = Embedding(embedding_matrix.shape[0],
                            embedding_matrix.shape[1],
                            weights=[embedding_matrix],
                            trainable=False)

    model = Sequential()
    model.add(embedding_layer)
    model.add(LSTM(200, recurrent_dropout=0.2, dropout=0.2, return_sequences=True))

    model.add(LSTM(200))
    model.add(Dense(nb_classes))
    model.add(Activation('softmax'))
    model.summary()

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    rnn = model.fit(X_train, y_train, nb_epoch=nb_epoch, batch_size=batch_size, shuffle=True,
            validation_data=(X_val, y_val))
    score = model.evaluate(X_val, y_val)
    print("Test Loss: %.2f%%" % (score[0] * 100))
    print("Test Accuracy: %.2f%%" % (score[1] * 100))


    y_pred = model.predict (X_test)
    print ("\n\n\n\n\nY_Pred --> {}\n\n\n\n\n".format (y_pred))


    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy', f1_m,
precision_m, recall_m])


    history = model.fit(X_train, y_train, validation_split=0.3, epochs=10, verbose=0)


    loss, accuracy, f1_score, precision, recall = model.evaluate(X_test, y_test, verbose=0)

    print ("\n\n\n Loss -> {}\nAccuracy -> {}\nF1_score -> {}\nPrec -> {}\nRecall -> {}\n".format
(loss, accuracy, f1_score, precision, recall))


    model_name = "/home/san1234/thesis/sampling_temp.model"
    model.save(model_name)
    print ("\n\n\nMODEL SAVED SUCCESSFULLY !!!\n\n\n")

def recall_m (y_true, y_pred):
    """ Calculate the recall value """

def precision_m (y_true, y_pred):
    """ Calculate the precision value """

def f1_m (y_true, y_pred):
    """ Calculate the F-Score """

def process_reviews(df_reviews):

    """ This function does the pre-processing of raw data """

def clean_reviews(review):

   """ This function cleans each line of text data """

def process_labels(ls_labels):

   """ This function is used to make the 5 classes into binary class """

if __name__ == '__main__':
    preprocess()
```

10. **Neural_Net_Word2vec**

```
""" IMPORTING REQUIRED PYTHON MODULES """

def preprocess():
    df = pd.read_csv('Coursera_Reviews.csv')
    dframe = pd.DataFrame()
    dframe['Review'] = df['Review']
    dframe['Label'] = df['Label']

    print (dframe.columns)

    df_reviews = (dframe ['Review'])
    df_labels  = (dframe ['Label'])

    main_dframe = pd.DataFrame ({'Reviews' : ls_reviews, 'Labels' : ls_labels})
    print (main_dframe)

    num_features = 300
    min_word_count = 1
    num_workers = 4
    context = 10
    downsampling = 1e-3

    print ("\nTraining model.... \n\n")

    model = word2vec.Word2Vec (main_dframe ['Reviews'],
                               workers = num_workers,
                               size = num_features,
                               min_count = min_word_count,
                               window = context,
                               sample = downsampling)
```

```
model.init_sims (replace=True)

model_name = "/home/san1234/thesis/sampling.model"
model.save(model_name)

num_features = 300

top_words = w_model.wv.syn0.shape[0]
mxlen = 50
nb_classes = 3

train_set, test_set = train_test_split (main_dframe, test_size=0.2)
batch_size = 128
nb_epoch = 1
n_timesteps = 10

embedding_layer = Embedding(embedding_matrix.shape[0],
                  embedding_matrix.shape[1],
                  weights=[embedding_matrix],
                  trainable=False)

model = Sequential()
model.add(embedding_layer)
model.add(LSTM(200, recurrent_dropout=0.2, dropout=0.2, return_sequences=True))

model.add(LSTM(200))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))
model.summary()

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
rnn = model.fit(X_train, y_train, nb_epoch=nb_epoch, batch_size=batch_size, shuffle=True,
        validation_data=(X_val, y_val))
score = model.evaluate(X_val, y_val)
```

```
    print("Test Loss: %.2f%%" % (score[0] * 100))
    print("Test Accuracy: %.2f%%" % (score[1] * 100))


    y_pred = model.predict (X_test)
    print ("\n\n\n\n\nY_Pred --> {}\n\n\n\n\n".format (y_pred))

    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy', f1_m,
precision_m, recall_m])


    history = model.fit(X_train, y_train, validation_split=0.3, epochs=10, verbose=0)


    loss, accuracy, f1_score, precision, recall = model.evaluate(X_test, y_test, verbose=0)

    print ("\n\n\n Loss -> {}\nAccuracy -> {}\nF1_score -> {}\nPrec -> {}\nRecall -> {}\n".format
(loss, accuracy, f1_score, precision, recall))


    model_name = "/home/san1234/thesis/sampling_temp.model"
    model.save(model_name)
    print ("\n\n\nMODEL SAVED SUCCESSFULLY !!!\n\n\n")

def recall_m (y_true, y_pred):
    """ Calculate the recall value """


def precision_m (y_true, y_pred):
    """ Calculate the precision value """


def f1_m (y_true, y_pred):
    """ Calculate the F-Score """
```

```
def process_reviews(df_reviews):

    """ This function does the pre-processing of raw data """

def clean_reviews(review):

   """ This function cleans each line of text data """

def process_labels(ls_labels):

   """ This function is used to make the 5 classes into binary class """

if __name__ == '__main__':
    preprocess()
```

11. **Bi-LSTM_Algorithm**

```
""" IMPORTING ALL THE REQUIRED PYTHON MODULES """

march8 = pandas.read_csv('email_data_200', encoding="ISO-8859-1")

ls1 = []
ls2 = []



march8 = march8.sort_values(['Case Number'], ascending=[True])
march8 = march8.dropna(subset=['Text Body'])
march8 = march8.reset_index(drop=True)
march8.head(30)

def read_email(fname):
    """ This is only a testing function which reads the email text data as a unicode """

def corpus2sentences(corpus):
    """ This function splits a corpus into a list of sentences """



case_dict = {}
new_set = [] #dictionaries are stored in this before they are added to a dataframe
case_nos = list(march8['Case Number'])[:4] #get the first n case numbers, so n threads
count = 0
for case in case_nos:
  count += 1
  print(count)
  random_thread = march8.loc[march8['Case Number'] == case]
  message_list = list(random_thread['Text Body'])

  email_counts = []
  split_on                                                                                    =
"From:(?:[^\n]+)\nSent:(?:[^\n]+)\nTo:(?:[^\n]+)\n(?:Cc:(?:[^\n]+)\n)?Subject:(?:[^\n]+)\n"

  """ Steps of pre-processing of raw data """

apos = {
"aren't" : "are not",
"can't" : "cannot",
"couldn't" : "could not",
"didn't" : "did not",
"doesn't" : "does not",
"don't" : "do not",
"hadn't" : "had not",
"hasn't" : "has not",
"haven't" : "have not",
"he'd" : "he would",
"he'll" : "he will",
"he's" : "he is",
"i'd" : "I would",
"i'd" : "I had",
"i'll" : "I will",
"i'm" : "I am",
"isn't" : "is not",
"it's" : "it is",
"it'll":"it will",
"i've" : "I have",
"let's" : "let us",
"mightn't" : "might not",
"mustn't" : "must not",
"shan't" : "shall not",
"she'd" : "she would",
"she'll" : "she will",
"she's" : "she is",
"shouldn't" : "should not",
"that's" : "that is",
"there's" : "there is",
"they'd" : "they would",
"they'll" : "they will",
"they're" : "they are",
"they've" : "they have",
"we'd" : "we would",
"we're" : "we are",
"weren't" : "were not",
"we've" : "we have",
"what'll" : "what will",
"what're" : "what are",
"what's" : "what is",
"what've" : "what have",
"where's" : "where is",
"who'd" : "who would",
"who'll" : "who will",
"who're" : "who are",
"who's" : "who is",
"who've" : "who have",
"won't" : "will not",
"wouldn't" : "would not",
"you'd" : "you would",
"you'll" : "you will",
"you're" : "you are",
"you've" : "you have",
"'re": " are",
"wasn't": "was not",
"we'll":" will",
"didn't": "did not"
}

#remove all the standard stopwords that seem to contain sentiment - especially those involving
```

```
"not"
custom_stopwords = ['rt','i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'between', 'into', 'through', 'during', 'before', 'after', 'to', 'from', 'in', 'out', 'on', 'off', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ma']

def text_process(df):
  """ This function cleans each row of text data """



def process(df, ReviewCol, RatingCol):
  new_df = pd.DataFrame(
    {'Review': df[ReviewCol],
     'Rating': df[RatingCol]
    })
  return new_df

X_test = test_sentences
top_words = 51853#embedding_matrix.shape[0]
mxlen = 50 #tested, this is about right
nb_classes = 3

with open('/home/san1234/dask/sentiment_analysis/X_train_yuge.pkl', 'rb') as f:
  X_train = pickle.load(f)
tru_neu = 0
tru_pos = 0
tru_neg = 0 #number of true positives and negatives ENCOUNTERED
classified_neu = 0
classified_pos = 0
```

```
classified_neg = 0 #for the ones we actually classified (didn't ignore), sum up these three
corr_pos = 0
corr_neg = 0
corr_neu = 0 #for overall accuracy, sum the last 3 and divide by the number of reviews
rev = list(basic['Message Body'])
#rat = list(combo_vader['Rating'])
for i in range(len(rev)):
  #print(review, "\n\n")
  review = rev[i]
  score = sid.polarity_scores(review)
  pos_score = score['pos']
  neu_score = score['neu']
  neg_score = score['neg']
  comp_score = score['compound']


  if pos_score > 0.2 and neg_score < 0.2:
    classified_pos += 1
    y_classes[i] = 1



  elif neg_score > 0.2 and pos_score < 0.2:
    classified_neg += 1
    y_classes[i] = -1

  elif neu_score > 0.75:
    classified_neu += 1
    y_classes[i] = 0

test_list = list(basic['Message Body'])[:10]
for index, item in enumerate(test_list):
  print(y_classes[index])
  print(item)
```

```
df = pd.DataFrame()
df ['Text Body'] = march8 ['Text Body']
df ['Labels'] = march8 ['Labels']

max_words = 20000
print ("LENGTH ===> {}".format(df))



train_x = df ['Text Body'] [0:75800]
test_x  = df ['Text Body'] [75801:108285]

train_y = df ['Labels'] [0:75800]
test_y  = df ['Labels'] [75801:108285]

print ("Creating Bi-LSTM model... ")
e_init = K.initializers.RandomUniform(-0.01, 0.01, seed=1)
init = K.initializers.glorot_uniform(seed=1)
simple_adam = K.optimizers.Adam()
embed_vec_len = 32



model = K.models.Sequential()
model.add(K.layers.embeddings.Embedding(input_dim=max_words,  output_dim=embed_vec_len,
embeddings_initializer=e_init,mask_zero=True))
model.add(K.layers.LSTM(units=100, kernel_initializer=init, dropout=0.2, recurrent_dropout=0.2))
model.add(Bidirectional(LSTM(20, return_sequences=True), input_shape=(n_timesteps, 1)))
model.add(K.layers.Dense(units=1, kernel_initializer=init, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer=simple_adam, metrics=['acc'])

bat_size = 32
max_epochs = 2
print ("\nStarting training... ")
```

```
model.fit(train_x, train_y, epochs=max_epochs, batch_size=bat_size, verbose=2)
print ("Training complete... \n")

loss_acc = model.evaluate (test_x, test_y, verbose=2)

print("Test data: loss = %0.6f  accuracy = %0.2f%% " % (loss_acc[0], loss_acc[1] * 100))

time.sleep (2.0)
print ("COMPLETE... ")
```

12. **Algorithm_IMDB**

```
“”” IMPORTING PYTHON MODULES “””
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

def main():
  print("IMDB sentiment analysis using Keras/TensorFlow")
  np.random.seed(1)
  tf.set_random_seed(1)

  max_words = 20000
  print("Loading data, max unique words = %d words\n" % max_words)
  (train_x, train_y), (test_x, test_y) = K.datasets.imdb.load_data(seed=1, num_words=max_words)

  max_review_len = 80
  train_x = K.pre-processing.sequence.pad_sequences(train_x, truncating='pre', padding='pre',
maxlen=max_review_len)
  test_x = K.pre-processing.sequence.pad_sequences(test_x, truncating='pre', padding='pre',
maxlen=max_review_len)

  print("Creating LSTM model")
  e_init = K.initializers.RandomUniform(-0.01, 0.01, seed=1)
  init = K.initializers.glorot_uniform(seed=1)
  simple_adam = K.optimizers.Adam()
  embed_vec_len = 32  # values per word

  model = K.models.Sequential()
  model.add(K.layers.embeddings.Embedding(input_dim=max_words,
output_dim=embed_vec_len, embeddings_initializer=e_init, mask_zero=True))
  model.add(K.layers.LSTM(units=100, kernel_initializer=init, dropout=0.2,
recurrent_dropout=0.2))  # 100 memory
  model.add(K.layers.Dense(units=1, kernel_initializer=init, activation='sigmoid'))
  model.compile(loss='binary_crossentropy', optimizer=simple_adam, metrics=['acc'])
  print(model.summary())

  bat_size = 32
  max_epochs = 3
  print("\nStarting training ")
  model.fit(train_x, train_y, epochs=max_epochs,
  batch_size=bat_size, shuffle=True, verbose=1)
  print("Training complete \n")

  loss_acc = model.evaluate(test_x, test_y, verbose=0)
  print("Test data: loss = %0.6f  accuracy = %0.2f%% " % \
  (loss_acc[0]*100, loss_acc[1]*100))

#--------------------------------

def main_trial():
  print("IMDB sentiment analysis using Keras/TensorFlow")
  np.random.seed(1)
  tf.set_random_seed(1)

  max_words = 20000
  max_review_len = 80
  df = pandas.read_csv ("cloth_short.csv")
  df = df.dropna()

  train_x = df['Reviews'][0:140]
  test_x  = df['Reviews'][141:197]

  train_y = df['Labels'][0:140]
  test_y  = df['Labels'][141:197]

  train_x = K.pre-processing.sequence.pad_sequences(train_x, maxlen=max_review_len,
dtype='int32', padding='pre', truncating='pre', value=0.0)
```

```
  test_x = K.pre-processing.sequence.pad_sequences(test_x, maxlen=max_review_len, dtype='int32', padding='pre', truncating='pre', value=0.0)

  print("Creating LSTM model")
  e_init = K.initializers.RandomUniform(-0.01, 0.01, seed=1)
  init = K.initializers.glorot_uniform(seed=1)
  simple_adam = K.optimizers.Adam()
  embed_vec_len = 32  # values per word

  model = K.models.Sequential()
  model.add(K.layers.embeddings.Embedding(input_dim=max_words, output_dim=embed_vec_len, embeddings_initializer=e_init, mask_zero=True))
  model.add(K.layers.LSTM(units=100, kernel_initializer=init, dropout=0.2, recurrent_dropout=0.2))  # 100 memory
  model.add(K.layers.Dense(units=1, kernel_initializer=init, activation='sigmoid'))

  model.compile(loss='binary_crossentropy', optimizer=simple_adam, metrics=['acc'])
  print(model.summary())

  bat_size = 32
  max_epochs = 3
  print("\nStarting training ")
  model.fit(train_x, train_y, epochs=max_epochs, batch_size=bat_size, shuffle=True, verbose=1)
  #model.fit (train_x, train_y, epochs=max_epochs, batch_size=bat_size)
  print("Training complete \n")

  loss_acc = model.evaluate(test_x, test_y, verbose=0)
  print("Test data: loss = %0.6f  accuracy = %0.2f%% " % \
  (loss_acc[0], loss_acc[1]*100))

  #-----------------------------

if __name__ == '__main__':
  main()
```

Appendix B

This part of the appendix gives a brief explanation of various modules, algorithms and concepts which are made use of in this thesis.

1. Sentiment Analysis and Natural Language Processing (NLP)

The entire thesis is about Sentiment Analysis (SA). Sentiment Analysis is a field of study which concerns with opinions, sentiments, emotions or attitude of people towards entities like products, services, organizations or events and their attributes [97]. Sentiment analysis is also known by other names such as Opinion mining, Subjectivity analysis, Review mining etc [97].

Some applications of Sentiment Analysis include:

1. When a consumer wishes to make decisions about a product, important information about the reputation of that product is required. This is derived by people's opinions which Sentiment Analysis can reveal what people's opinions are [98].

2. Companies can know the opinions of the products they are selling [98].

3. Sentiment Analysis is proposed as a component of other technologies [98].

Classification of Sentiment Analysis:

The image above shows the classification of Sentiment Analysis based on three methods: Lexicon-based methods, Machine-learning methods and Hybrid methods.

Sentiment Analysis examines the problem of analysing texts such as reviews, comments or posts uploaded by people on social networking sites and use this data to judge the content as a positive or a negative one [99]. The entire thesis and the research which are documented in the previous chapters all focus on sentiment analysis on text data. The definitions mentioned above along with the classification make it clear as to what we are trying to achieve by performing sentiment analysis.

Natural Language Processing (NLP): NLP can be defined as technology which can adopt computers to interpret the natural language of Humans [101]. Some tasks included in NLP are Part-of-speech tagging, Chunking, Named-Entity-Recognition and others [102]. Researchers intend to gain knowledge about how humans understand and make use of language to develop appropriate tools and methods to enable computers to understand natural languages just like humans [103]. NLP has found its application in processing text data in a verity of languages such as Chinese, Hindi, Spanish and others [104]. Sentiment Analysis is a subset of Natural Language Processing.

2. Machine learning methods

We have seen various algorithms in the chapters concerning the research. Each algorithm is used to define a model developed. These chapters show algorithms in a step-by-step procedure right from a very basic model all the way till a very advanced algorithm which makes use of bidirectional Neural Networks. All of these algorithms perform sentiment analysis and is used to classify or train the data. This can be performed in three major ways.

1. Unsupervised learning: A category of machine learning which learns from test data and does not include labelled, classified or categorized data [105]. In this method of learning, the data can be grouped according to similarities and differences even if there are no categories provided [106]. Algorithm_1a, Algorithm_1b and Algorithm_2a make use of unlabelled data sets and the Textbook function provided by Python3.6. The algorithms classify the sentiment of data into positive, negative or neutral. This is an example of unsupervised learning. Unsupervised learning algorithms can be grouped into either clustering or association problems [107]. Clustering problem is the one which involves discovering the inherent groupings of data [107]. Association learning is the one which involves knowing the rules which describe large part of the data [107]. Some unsupervised learning algorithms include K-means clustering and Agglomerative clustering [108].

2. Supervised learning: This type of machine learning provides tools to classify and processes data which is labelled and which is classified [109]. The data set is used for predicting the classification of unlabelled data with the use of machine

learning algorithms [109]. Algorithm_2b onwards, we see the models making use of supervised learning. In these, we have labelled and classified data sets. An advantage of supervised learning over unsupervised learning is that we can know the number of classes there are in order to train the data [110]. Some supervised learning algorithms include Support Vector Machine, Naive Bayes, Neural Networks, Logistic Regression and others as we can observe in the image [100].

3. Reinforced learning: This involves the training of machine learning models to make a sequence of decisions [111]. In reinforced learning, a situation is encountered where the computer employees train and error to form a solution to a problem [111]. This thesis does not include reinforced learning.

3. Different Classification Machine learning Algorithms

The algorithms which are shown in the form of Python programs presented make use of one or more machine learning algorithms to perform classification and training. In Algorithm_2b and Algorithm_3a we see the use of K-Nearest-Neighbor (KNN) algorithm. In Algorithm_3c and Algorithm_3d we see three other classification algorithms Logistic Regression (log reg), Random Forest Classifier (RF) and Support Vector Machine (SVM). The reasons why these four algorithms were chosen are mentioned in an earlier section of the thesis. In this section, let us look at a detailed understanding of each of these algorithms, their applications, working and their advantages and drawbacks.

1. K-Nearest-Neighbors Algorithm (KNN): This algorithm is easy, simple to

implement supervised machine learning algorithm which can be used for classification and regression problems but is more useful for classification problems[138]. In order to evaluate a technique, three important aspects are looked into. The ease to interpret the output, Calculation time and Predictive power [139]. Among the three aspects, KNN is mainly used for the ease of interpretation of output and low calculation time [139].

Working of KNN algorithm:

The first step when using KNN algorithm is to transform data points into feature vectors or mathematical values [140]. The next step is that it finds the distance between the mathematical values of the points. The most common way to find the distance is by using the Euclidean distance [140].

$$\begin{aligned} d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \end{aligned} \quad (1)\text{--}$$

Shown above is the Euclidean distance formula [140]. KNN runs the above formula (1) to compute the distance between each data point and the test data. The probability of the points being similar to the test data is calculated and it is classified based on which points share the highest probabilities [140].

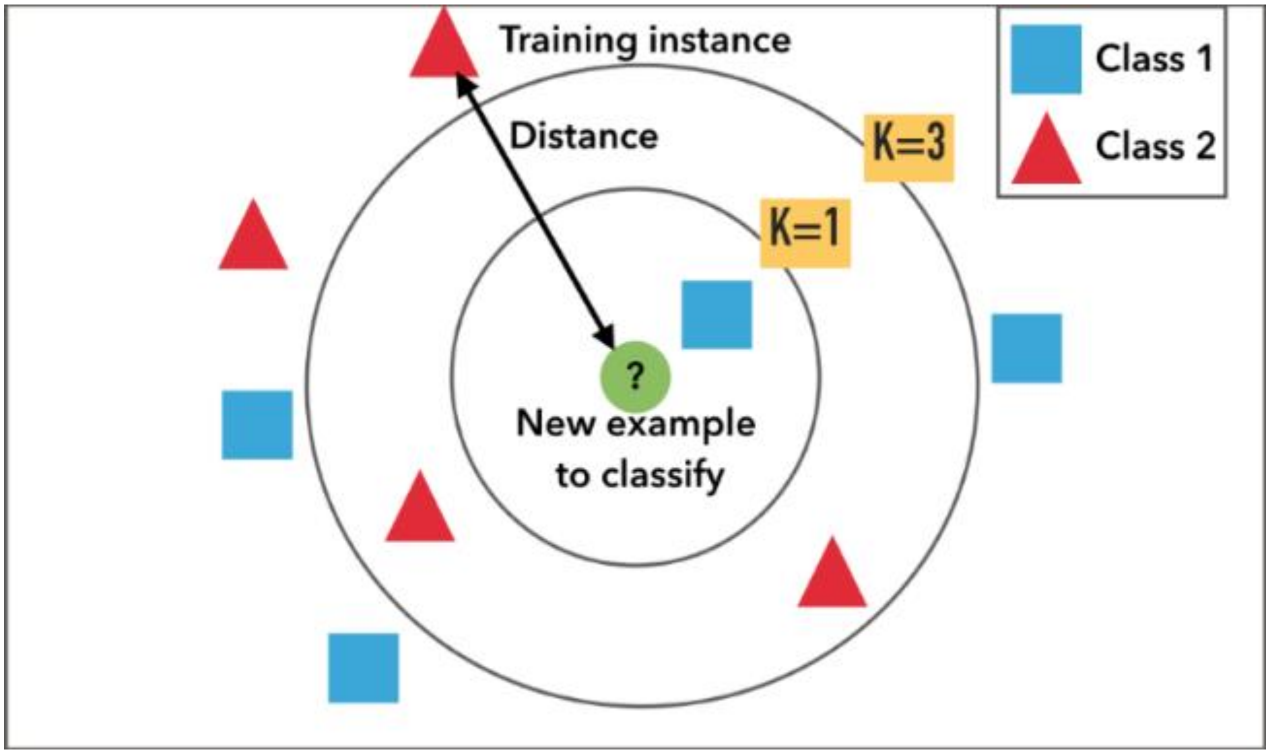


Image showing the KNN classification [141]

The above image shows the working of KNN algorithm using pictorial presentation. In the above image K is the number of neighbours. The test sample is represented by the inner circle. It is to either be classified into the first class represented by the blue squares or the second class represented by the red triangles. If we take the value of k = 3, then it is assigned to the second class because two red triangles are present and only one blue square is in the inner circle. If we assume k = 5, then it is assigned to the first class as three blue squares are present versus two red triangles in the outer circle [141].

Advantages of KNN algorithm:

1. Easy to implement [17].
2. The algorithm makes no pre assumptions about the data [17].

Drawbacks of KNN:

1. It requires homogeneous features [142].
2. One of the biggest issues is that it requires optimal number of neighbors [142].
3. It provides no good facility to treat missing values [142].

2. Logistic Regression Algorithm (log reg): This is a machine learning algorithm mainly used in case of binary classification problems [144]. Log reg algorithm is similar to the Linear regression algorithm but while the former is used to predict values, the latter is used to classify data [145]. Binary classification is where this algorithm finds its best applications where for example, classifying if a website is fake or not or checking if an email is spam or not is done [145]. Log reg algorithm uses a linear equation to predict values [145] since it is required to output the algorithm to be either a 0 or 1 for no and yes respectively [145]. For this, a Sigmoid function is made use of:

$$z = \theta_0 + \theta_1 \cdot x_1 + \theta \cdot x_2 + \cdots \quad \text{(1)--}$$

$$g(x) = \frac{1}{1 + e^{-x}} \quad \text{-- (2)}$$

$$h = g(z) = \frac{1}{1 + e^{-z}} \quad (3)\text{--}$$

In the above equations, ‘z’ represents the output of the linear equation. This output z is given to the function g(x) in equation (2) which returns a squashed value ‘h’. The value of h will be in the range of 0 to 1.

In the Algorithms in this thesis which make use of Logistic regression algorithm to classify data, the Sigmoid function will take into account the number of classes. The data set used for those algorithms have 5 classes ranging from 1 to 5. But irrespective of the number of classes, the output will be in a binary form, which is either 1 or 0.

Advantages of Logistic Regression:

1. It does not require many computational resources [146].
2. It does not require input features to be scaled [146].
3. It is efficient [146].

Drawbacks of Logistic Regression:

1. If there is a feature that would separate the two classes, Log reg model cannot be trained [147].
2. Log reg cannot solve non-linear classification problems since its decision surface is linear [148].

3. Random Forest Classifier algorithm (RF): Random forest algorithm is a supervised classification algorithm which creates a forest using a number of trees [149]. It is mentioned that when there are more trees in the forest, the forest looks more robust and therefore, higher number of trees in the forest can give better accuracy values [149]. This algorithm is built on top of the Decision tree algorithm. This is one of those algorithms which can be used for both classification and regression tasks [149]. Although this thesis mainly focuses on classification of data.

Let us look into an example to understand the RF algorithm even better. Let us consider a node which has to make a decision as to which channel out of many it has to send a message. It looks into the history of another node and gets information about the channels it has used and based on this makes a decision. This is analogous to the Decision Tree algorithm. If the first node feels that the information from the other node could be biased, it will look into the history of several other nodes and come up with a better decision. This is Random Forest algorithm.

Working of the RF algorithm: There are two major stages in the RF algorithm. The first one deals with the creation of a forest and the second one is to make a prediction. First, “K” features are selected from a total of “m” features. Next, among the “K” features, the best split point is used to calculate the node “d”. The node is then split into number of nodes using best split. The above three steps are repeated until a certain number of nodes are reached. To create “n” number of trees, the above four steps have to be repeated for "n" number of times.

Advantages of Random Forest algorithm:

1. This algorithm avoids the over fitting problem [150]. Over fitting is a problem where the algorithm continues to develop hypothesis which reduce the training error at the cost of increasing the testing error [151].
2. The same algorithm can be used for both classification and regression tasks [150].
3. It can be used to identify the most intricate features from the training data set [150].

Drawbacks of Random Forest algorithm:

i) The main drawback is the algorithm's complexity. This occurs due to the creation of many trees and combining the outputs [152].

ii) Takes more time for training [152].

4. Support Vector Machine algorithm (SVM): This algorithm is based on the concept of decision planes which define a boundary [153]. A decision plane separates between a set of objects having different class memberships [153]. SVM model represents the examples as points in space mapped in such a way that the examples of separate categories are divided by a clear gap [154].

Working of SVM algorithm:

The SVM model creates a hyper-plane which separates data into classes. It puts a theoretical line or a hyper-plane that separates every class [156]. It takes data as input and gives the line that separates those classes as outputs [156].

In the Appendix-D section, image_08 shows an example in which there are two classes [156]. One class is represented by the red squares and the other class is represented by blue circles. The goal is to find an ideal line that separates the two classes. But we can observe more than one line on the image above. There are two lines. So the ambiguity is to know which is better to choose. The algorithm chooses the yellow line as the hyper-plane. The green line although separates the two classes, it is not generalized line.

Advantages of SVM [156]:

1. It works well when there is a clear margin of separation between classes.

2. It is efficient in cases where a number of dimensions is greater than the number of samples.

3. It is memory efficient.

Drawbacks of SVM [156]:

1. It is not suitable for large data sets.

2. When the data is noisy, this algorithm fails to perform well.

3. The algorithm will underperform where the number of features for each data point exceeds the number of training samples.

4. Performance Metrics

The thesis deals with how I conducted my research to answer the research question which deals with performing sentiment analysis on large text data. We have come up with three models two of which are for text data set with comment data and one for a very large email data set. All of these algorithms involve the calculation of four parameters. Precision, Recall value, F-score and Accuracy. Let us discuss in brief what these values are. But before that let us look into some definitions:

1. True Positive: It is an outcome where the model predicts correctly the positive class [157].

2. True Negative: It is an outcome where the model predicts correctly the negative class [157].

3. False Positive: It is an outcome where the model incorrectly predicts the positive class [157].

4. False Negative: It is an outcome where the model incorrectly predicts the negative class.

Let us take an example to explain the above definitions. In case of a game of cricket, when the umpire is to make a decision whether the batsman is out or not,

* Umpire gives NOT OUT when the batsman is not out: True Positive.

* Umpire gives OUT when the batsman is out: True Negative.

* Umpire gives NOT OUT when the batsman is out: False Positive.

* Umpire gives OUT when the batsman is not out: False Negative.

Now that we have learnt about the above four definitions, let us look into the parameters and what they mean:

1. Precision: Precision gives us the value of what propagation of the positive identifications were actually correct [158].

Precision = TP / (TP + FP)   --- (1)

Where: TP — True Positive   FP — False Positive

2. Recall: Recall gives us the value of what propagation of actual positives were identified correctly [158].

Recall = TP / (TP + FN)   --- (2)

Where: TP — True Positive   FN — False Negative

3. Accuracy: Accuracy is the fraction of predictions which the model has got right [158].

Accuracy = (TP + TN) / (TP + TN + FP + FN)   --- (3)

Where: TP — True Positive	TN — True Negative
FP — False Positive	FN — False Negative

4. F1 Score: This gives a balance between the precision and recall [159].

F1score = 2 * (Precision * Recall) / (Precision + Recall) --- (4)

5. Recurrent Neural Networks

The research section has a chapter which explains Artificial Neural Networks (ANNs) in detail. The chapter also explains why ANNs was adopted to build a model, its advantages and disadvantages. Deep learning and Deep Neural Networks (DNNs) have also been explained in detail. In this section we shall take a look at Recurrent Neural Networks (RNNs) and its types.

1. Recurrent Neural Network (RNN): This is a class of Neural Networks which enable the previous outputs to be used as inputs in the form of hidden states [160].

The above image [160] shows an architecture of a Recurrent Neural Network where the output of the previous node is the input to the next one.

Advantages of RNN [160]:

- The possibility of processing inputs of any length is high.
- Weights are shared across time.
- Historical information is considered for computation.

Drawbacks of RNN [160]:

- Computation is slow.
- Accessing information from a long time in the past is a problem.
- Future input cannot be considered for the current state.

2. Bidirectional Recurrent Neural Network (Bi RNN): Bi RNNs connect two hidden layers in opposite directions to a single output which allow them to receive information from both past and future states [161]. This deep learning technique is more prominent is supervised learning approaches [161].

The image above shows the structure of a bidirectional RNN [162].
From this structure we can infer that each output is given as an input to another node and at the same time, it receives information from both the previous node and the next node.

3. Bidirectional Long Short Term Memory (Bi LSTM): We have already seen the working of LSTM. Here we shall briefly observe the working of a bidirectional LSTM. Bi LSTM incorporates the concepts of Bi RNNs and LSTM and connects two hidden LSTM layers in opposite directions to a single output. This way, it is able to receive information from both past and present and is capable of retaining the memory for a long time. Algorithm_email_SA makes use of Bidirectional LSTM.

Appendix C

This part of the appendix explains the tools, and Python modules used to develop the algorithms.

1. Python 3.6 and its modules

In order to implement the models and algorithms, the programming language made use of is Python3.6. Python is an interpreted object oriented programming language created in 1990 which differs from other programming languages wherein it marked code readable and use white space over compact small source files [112]. Python3.6 is one of the latest versions of the programming language and as we have seen in the algorithms, offers a wide range of in built libraries and modules to perform various functions in domains of mathematics, machine learning ad others. Python is a high level language and is interpreted, which means, it is written in a compiled language (C or C++) using a compiler. Hence, Python does not require compilation. Let us take a look at some of the major Python libraries used in the algorithms:

A. Numpy: Numpy is a basic package provided by Python for scientific computing [113]. Numpy provides us with an object N-dimensional array (array) [114]. It also provides us with a universal function object (ufunc). An N-dimensional array is a collection of items which are indexed using integers [114].

In Algorithm_3d, we see the use of numpy. In this algorithm, Numpy is imported as ‘np’. This is the algorithm which represents the vectorization model. We see in line 108 numpy.mean() method being used. Numpy.mean() Returns the average of the array elements [115]. In Algorithm_NN1, we see the use of Numpy module where a functionality np.zeros() is used. This is observed in line 159. What this does is, it returns a new array of a shape and size which is filled with zeros [116]. We also observe in line 163 numpy.sum() function. This function adds the array elements and gives the sum of them [117].

B. Pandas: Pandas is a high level tool used for manipulation of data built on the Numpy package and provides Data Frame as a key data structure [118]. These data frames enable us to store and manipulate data in tabular form which are in the shape of rows and columns [118]. We can observe the use of a Pandas data frame in every algorithm used. These data frames not only provide the required tabular data structure to store our data in rows and columns but also provide us with functionalities with which we can change or operate on the data in the data frames.

In every algorithm, we can see that to create a data frame out of a.csv file, we write the following line:

```
dframe = pandas.read_csv (“Food_Reviews.csv”)
```

Here, dframe is the name of the data frame we intend to create, pandas.read_csv() is a function of pandas which will create a data frame from a.csv file and a string parameter is provided which contains the name of the csv file with which we want to create the data frame. We can also see in many of the machine learning algorithms how we can create a new row in a data frame, delete a row, select only a particular part of the rows and other functions.

C. Other libraries: Apart from Numpy and Pandas, there are other libraries which are used in the algorithms. Let us see some of them in brief:

‘Re’: This module provides regular expressions functionalities to match patterns [119]. We can see this module in all the machine learning algorithms used in the data pre-processing steps to remove punctuations from raw data as the punctuations are considered noisy. In the algorithm concerning Twitter sentiment analysis (Algorithm_2a), the re module is used to remove emotions as well.

‘String’: This module provides us with many useful constants using which we can operate on string data [120]. We can see the use of the ‘string’ module in the pre-processing steps in our algorithms.

‘sklearn’: This is the SciKit learn module which is already discussed before. It provides us various algorithms used for machine learning and also provides functions to perform numerical operations on data [121]. We can see several sklearn functions imported in our algorithms, and they are all used to perform classification on the data with a machine learning algorithm and to calculate numerical parameters like precision, recall and accuracy.

‘text table’: It is a simple library provided by Python to read and write ASCII text tables [122]. We have used this module to create tables to compare the parameters of different algorithms or readings.

2. Python NLTK (Natural Language Tool Kit)

Natural Language Toolkit or NLTK is an open source library provided by Python which provides us with tools and functions to enable us to perform operations on text data or to process natural language [7]. NLTK also finds its application in text analytics [8]. Let us take a look at some of the tools provided by NLTK which have been made use of in the algorithms:

A. Word and Sentence tokenizer: Tokenization is the process of breaking up a series of text data into chunks such as words or sentences called tokens [123]. Word tokenizing is to break up a series of strings into words where each word is called a token. The nltk.word_tokenize() returns a tokenized copy of the text [124].

An example: Assume a string str1 = “The cat will fit into the box”

```
Here str1 is a string.
Import nltk
word = nltk.word_tokenize (str1)
print (word)
```

The output: [“The”, “cat”, “will”, “fit”, “into”, “the”, “box”]

The same principle is seen in case of sentences where we use nltk.sent_tokenize().

In this case, the entire block of text is split into sentences each of which is called a token [125].

An example: Assume a string str2 = "There are clouds. It will rain tonight. So be prepared please"

```
            import nltk
      sent = nltk.sent_tokenize (str2)
      print (sent)
```

The output: ["There are clouds", "It will rain tonight", "So be prepared please"]

Word tokenize is used in the pre-processing steps of the machine learning algorithms.

B. Stop Words: The text data can contain words like 'is', 'to', 'the', 'on' etc. There are referred to as stop words and it is preferred to remove them from the raw data [126]. In the algorithms, if we look at the pre-processing steps, we can see the stop words being removed.

```
From nltk.corpus import stopwords = list (stop words.words ('English'))
print (stopWords)
```

The output will be a list of inbuilt stop words in Python. Using this list, we can filter out the stop words in a list. We can also add our own words into the list of stop words or if we wish remove one or more words.

C. Stemming: Stemmers extract the morphological affixes from a word which leaves only the word stem.

For example, if we have four words “activate”, “activity”, “activation”, “activation al”:

```
from nltk.stem import PorterStemmer
ps = PorterStemmer()
words = [“activate”, “activity”, “activation”, “activation al”]
for w in words:
        print (ps.stem (w))
```

```
Output: activ, activ, activation, activ
```

From the output, we can see that three out of four words have been stemmed down to their basic word ‘activ’. This way, if we can stem down similar words, it will be cleaner data for processing.

D. Named Entity Recognition: Named Entity recognition locates and classifies named entities in text into pre-defined categories such as person, location, organization percentages, currency etc [127]. Using NER tagging, we can know the names of companies mentioned in an article, or the proper nouns in a block of text [127].

For example, if we look into Algorithm_3b, we see how named entity recognition is used to tag words that belong to an entity and remove them. First, chunks of text are created with their entity tags into a variable namedEnt. Then an empty list is creates. The list namedEnt is iterated over and wherever we see a tag 'NNP' which corresponds to proper nouns, we remove them.

3. TextBlob

Textbook is a library provided by Python for processing textual data and provides simple APIs for performing Natural Language Processing (NLP) like part-of-speech tagging, noun phrase extraction and sentiment analysis [128]. Textbook is useful in many an industry where dealing with and processing large data comes into play [129]. We can see the use of Textbook in three of the algorithms in this thesis. Algorithm_1a, Algorithm_1b and Algorithm_2a. Textbook comes with many sub modules and functionalities. One of them is the text blob.sentiment.polarity() function which is made use of in the algorithms mentioned above. What this function does is, it gives us the polarity of an English phrase of group of words as positive ot negative [130]. We can observe in Algorithm_12 that the entire row of text data from the data set is put into a Python list. The list is then iterated where in each iteration one row of text data is taken in and then on this string of text data text blob.sentiment.polarity is used which gives it a polarity. Similarly, in Algorithm_2a we use the same technique. Here a word is taken as an input and the number of occupants of that word is taken as another input. A variable 'analysis' is created which represents a Textbook object and analysis.sentiment.polarity is used and conditions are given. If the polarity exceeds 0, it is positive. If the polarity is below 0, it is considered negative polarity and if the polarity is 0 it is neutral polarity. In an article which dealt with sentiment analysis to predict election results [131], Textbook was used and more specifically text blob.sentiment.polarity was used to predict the USA election results. Like NLTK, even Textbook comes with a word tokenizer which creates a chunk of words from text data. This word tokenizer of Textbook is shown in this paper [132]. Textblob can be used to analyse particular category of sentiment and classify a word into that category [137].

4. SciKit Learn

Scikit learn is a Python module which provides a range of selection of supervised and unsupervised learning algorithms [133]. Scikit learn is by far the easiest and cleanest machine learning library in Python [133]. Scikit Learn is built on top of some data and math libraries of Python [133] such as:

i) Numpy
ii) SciPy
iii) Matplotlib
iv) Ipython
v) Sympy
vi) Pandas

Defines Scikit learn as an open source library which provides a consistent API for using machine learning algorithms in Python [134]. The main goal of using Scikit learn is to simplify the contentions and to limit the number of methods an object should implement [134]. The name ‘Scikit learn’ is derived from SciPy ToolKit and in short is also referred to as sklearn [134].

In the algorithms presented in this thesis, Scikit learn is the most widely used and most frequently used Python library. Several functionalities of Scikit learn have been made use of. Right from Algorithm_2b to Algorithm_email_SA, we can see various functionalities of Scikit learn to be used.

Let us look into Algorithm_2b. In this algorithm, let us look at the function tweet_analysis(). The steps are as follows:

Step-1: A data frame is got by calling another function process().

Step-2: A label encoder object is created. (Label encoder will be explained shortly).

Step-3: The two columns of the data frame are split into two lists and each of them are further split into training and testing lists. Here we make use of the train_test_split function of Scikit learn.

Step-4: We reshape each of the lists to make them suitable for classification. For this the pre-processing module of Scikit learn is made use of. Where we use the values.reshape() function.

Step-5: K-Nearest-Neighbor algorithm is used to classify and train the data. Even the KNN and its functionalities are a part of the Scikit learn package. From this, the accuracy, precision and other values are calculated.

As we can see, Scikit learn comes with many functionalities which are made use of in several other algorithms presented in the thesis. Let us look at a few important functionalities:

1. train_test_split(): This functionality of Scikit learn splits arrays or matrices into random train and test subsets by considering the parameters given [135].

We can import the module as follows:
from sklearn.model_selection import train_test_split

Then we can use the function as follows:
x_train, x_test, y_train, y_test = train_test_split (X, Y, test_size=0.2)

Where X and Y are arrays or lists containing data and test_size is a parameter that takes in the amount of test data size to be split into. Here 0.2 means 20% of the array or list will be split into test and 80% will be training data.

2. sklearn.metrics: This module provides us with score functions and performance metrics for computation [136]. Using this, we can import metric values and calculate values of precision, recall and f-score.

The module is imported like this:
from sklearn.metrics import precision_score, recall_score, f1_score

Then we can use this to calculate precision score as:
total_precision_score = precision_score (y_pred, y_test, average = 'weighted')

Similarly, other values like recall and f-score can be calculated.

3. sklearn.linear_model: This module is used to implement generalized linear models which includes Ridge regression and Stochastic gradient descent related algorithms [138]. In some of the algorithms, the Logistic Regression algorithm is made use of. This is imported from sklearn.linear_model.

It is imported as:
from sklearn.linear_model import LogisticRegression

It is used this way:
log reg = LogisticRegression()

Scikit learn provides us with handy tools and can perform a variety of things like predicting consumer behavior etc [169].

5. Keras

Keras is a high level neural network API which runs on top of Tensor flow, CNTK or Theano and is written in Python [163]. Keras is a part of the Tensor flow core [164] which makes it a high level API. Keras provides seven data sets which can be loaded directly [164]. One of the data sets (imdb movie reviews data set) has been imported and used in Algorithm_keras.

Why use Keras: Keras provides us with a wide range of functionalities most of which can be used in machine learning [163]. Some strengths of Keras which make it a preferred API are:

1. It has a broad adoption in the industry [164].
2. It supports multiple back end engines [164].
3. It shortens the amount of code needed for execution.

Tensor flow is an open source platform for machine learning which has comprehensive tools and libraries to enable the building of machine learning powered applications [165]. Using Keras which is an API for Tensor flow makes writing of code easier and in turn helping us ease the process to perform sentiment analysis.

Layers of Keras comes inbuilt with many modules to make the process of machine learning easier. We have already seen that Keras provides us with seven inbuilt data sets which can be imported and used. Keras layers are the building block of keras models [166]. Layers in Keras are created using the layer_ function [166]. There are a number of different layers provided by Keras. Some of them are:

1. Core layers 2. Pooling layers 3. Recurrent layers
4. Normalization layers 5. Embedding layers

In Algorithm_NN01, we are using some of these layers. We are using:

1. LSTM layer 2. Dense layer and
3. Activation layer 4. Embedding layer

In the algorithm, a model object is first created. Using this, we can perform Keras functions. Model.add() is a function which enables us to add a new layer. The first layer to be added in the LSTM layer. This layer comes under the Recurrent layer category. This layer is used to implement an LSTM layer [163]. In the algorithm, we create a LSTM layer by using the add() function. And in it, we supply the parameters recurrent_dropout, dropout and return_sequences. Following this layer, another LSTM layer is added with no parameters. After the creation of the LSTM layers, a Dense layer and an Activation layer are added.

The first layer to be created here in the Embedding layer. This layer turns positive indexes into dense vectors of fixed size [163]. The layer is first created by supplying parameters and is then added using the add() function.

Dense layer comes under the category of core layers. The Dense layer is a regular dense Neural network layer [163].

Activation layer is also a subclass of core layers. This applies an activation function to an output [163].

A pooling layer reduces the image dimensionality by retaining the important patterns [82].

6. DASK

Dask is a flexible library provided by Python for performing parallel computational tasks [175]. Dask is a revolutionary tool for data processing [176]. When we use Numpy or Pandas with respect to large data, there are cases where the data may not fit into the RAM [176]. This is where Dask comes into play. Dask supports the Data frame of Pandas and array data of Numpy and is capable of either running them on the local computer or be scaled up to run on a cluster [176].

Working of Dask:
Dask works on the concept of parallel computing. This means less time taken for execution. In case of a computer, if we assume it to have a fixed number of processors (let’s say 4 processors or cores), Dask splits the process equally into four parts and each processor is running a part of the task. But this is not all. Dask also uses the concept of lazy loading. Let us take the case of a data frame. When we use the Pandas module to create the data frame, the entire data set is loaded

into the data frame at once [177] and this is known as eager evaluation [177]. Pandas is also designed to work on a single core [177]. Dask on the other hand does not load the entire data set at once. It splits the data into many chunks and only loads one at a time [178].

The image_02 in the Appendix-D section shows the time and memory taken by a pandas data frame to load a data set compared with the time and memory taken by a Dask data frame to load the same data set. The time is measured in seconds and the memory is measured in bytes. We can see that in both time and memory utilization, Dask has the upper hand. This is because of three reasons:

1. The data set is large (About 1.3 Gigabytes). As mentioned before, Dask has the upper hand when data to be handled is large.

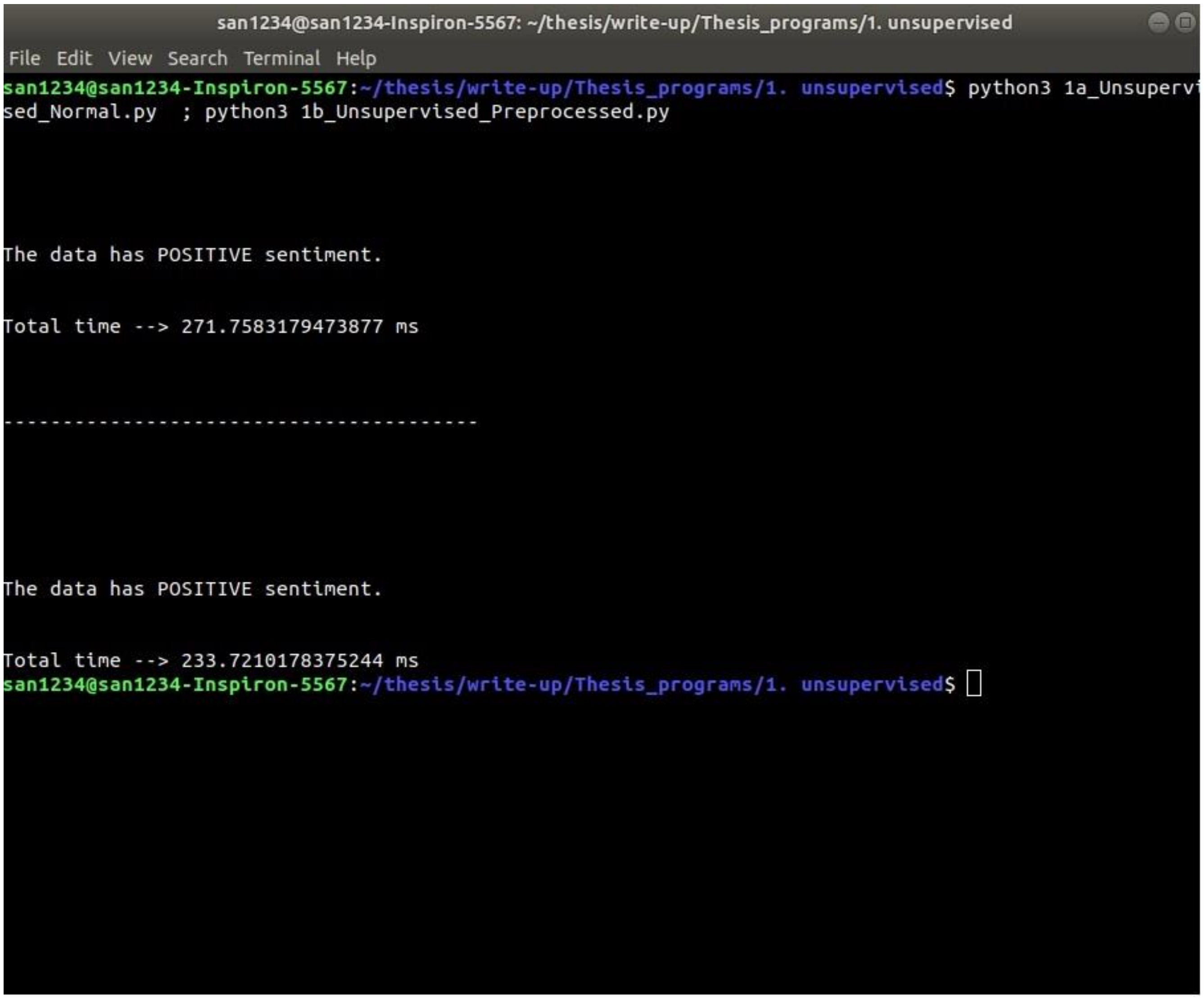
2. The time taken by Dask data frame is less because of the parallel computing abilities. It makes use of all the processors or cores of the system as opposed to just one core in the case of Pandas data frame.

3. The memory utilization is less in the case of Dask because of the lazy loading function. It splits the entire data into chunks and loads only one at a time.

A Dask data frame is a parallel data frame which comprises many smaller Pandas data frames [83]. Dask clusters can run on one single machine or on many remote networks [84]. Dask array provides chunked algorithms which work on top of libraries similar to Numpy [85].

Appendix D

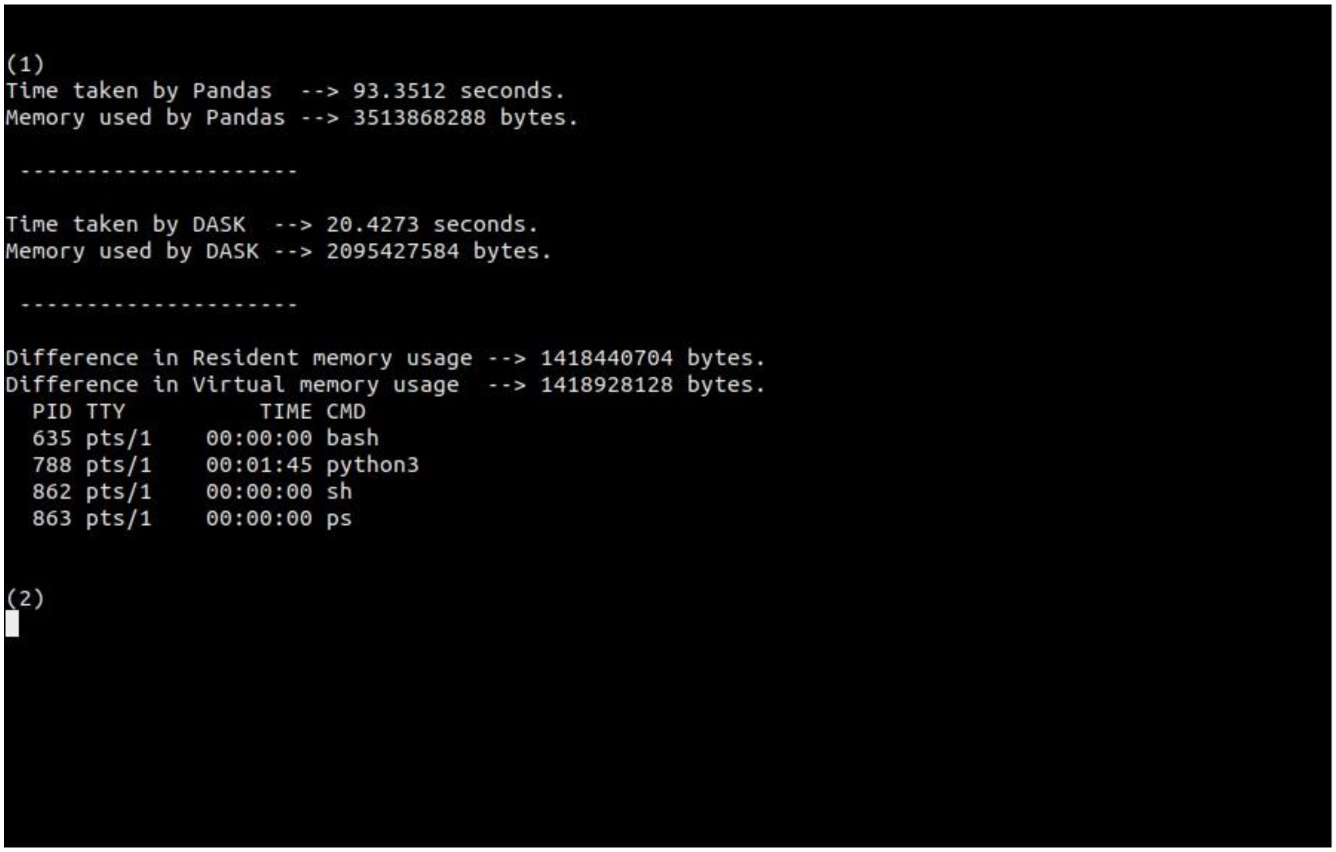
This part of the appendix contains images which are referred to in other sections like the methodology section.



san1234@san1234-Inspiron-5567: ~/thesis/write-up/Thesis_programs/1. unsupervised
File Edit View Search Terminal Help

```
san1234@san1234-Inspiron-5567:~/thesis/write-up/Thesis_programs/1. unsupervised$ python3 1a_Unsupervi
sed_Normal.py  ; python3 1b_Unsupervised_Preprocessed.py




The data has POSITIVE sentiment.


Total time --> 271.7583179473877 ms



----------------------------------------






The data has POSITIVE sentiment.


Total time --> 233.7210178375244 ms
san1234@san1234-Inspiron-5567:~/thesis/write-up/Thesis_programs/1. unsupervised$
```

Image_01: This image compares the time taken by Algorithm_1a versus the time taken by Algorithm_1b for execution.



```
(1)
Time taken by Pandas  --> 93.3512 seconds.
Memory used by Pandas --> 3513868288 bytes.

 --------------------

Time taken by DASK  --> 20.4273 seconds.
Memory used by DASK --> 2095427584 bytes.

 --------------------

Difference in Resident memory usage --> 1418440704 bytes.
Difference in Virtual memory usage  --> 1418928128 bytes.
  PID TTY          TIME CMD
  635 pts/1    00:00:00 bash
  788 pts/1    00:01:45 python3
  862 pts/1    00:00:00 sh
  863 pts/1    00:00:00 ps


(2)
```

Image_02: The comparison of time taken and memory used by pandas versus Dask.

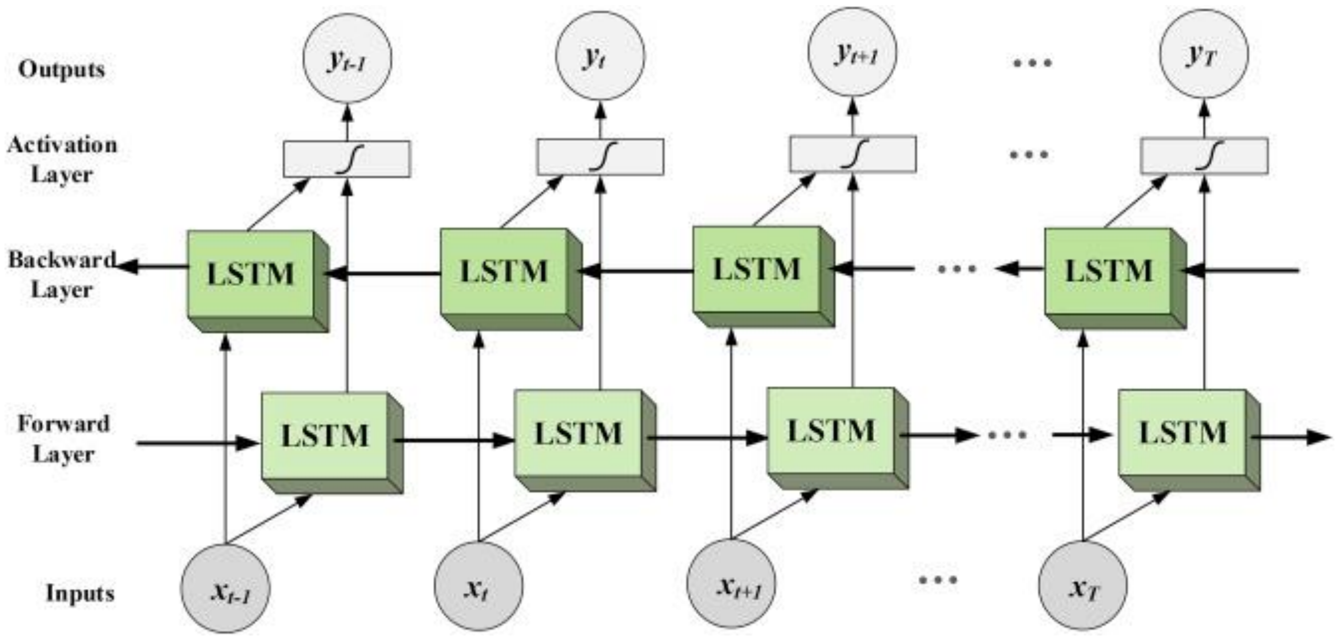


Image_03: This picture shows the structure of LSTM [174].

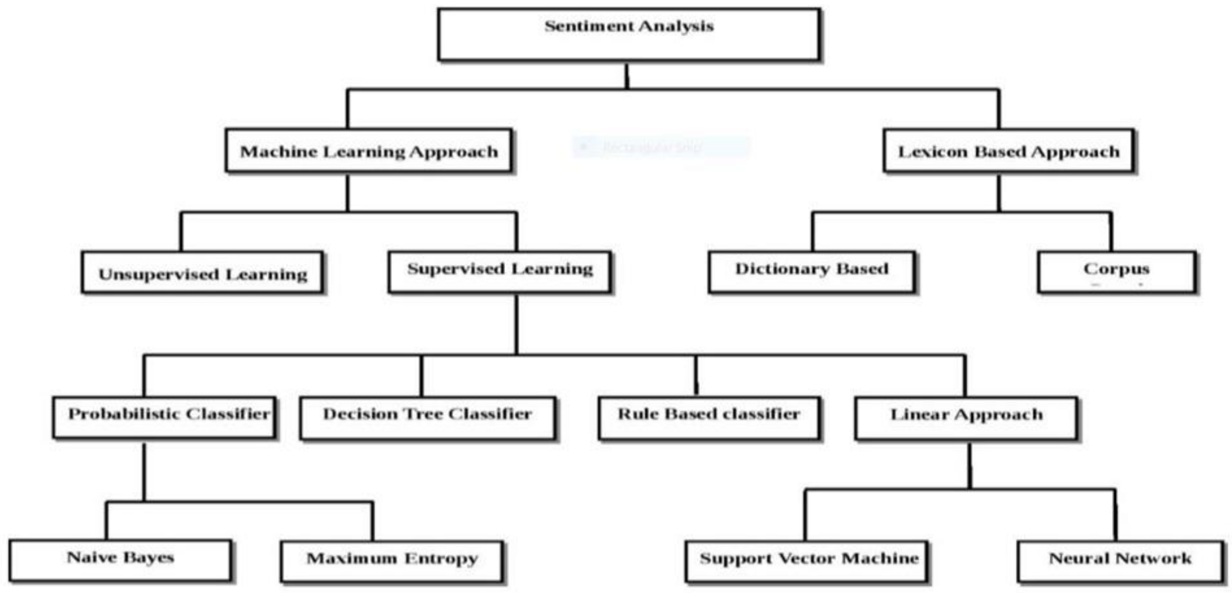


Figure1: Sentiment Classification Techniques

image_04: The image above shows a structure of classification of sentiment analysis [180].

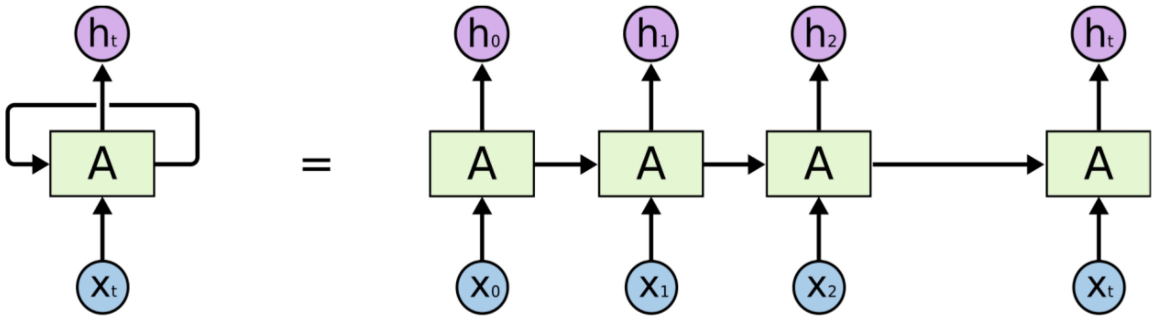


Image_05: The basic structure of a Recurrent Neural Network [47].

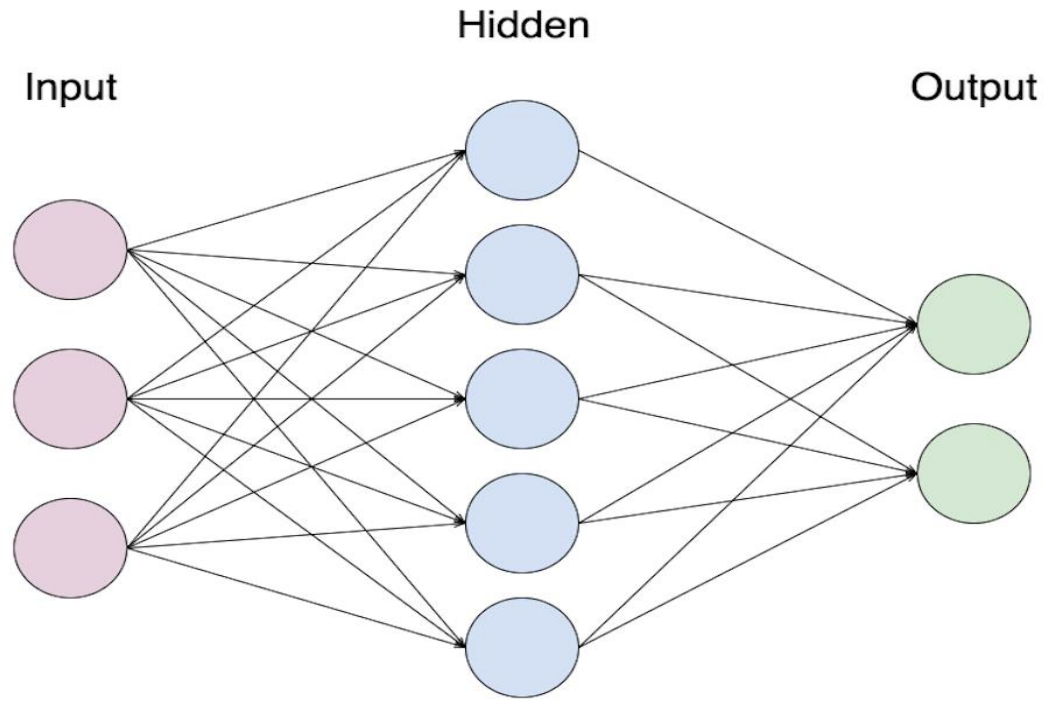


Image_06: Basic structure of an Artificial Neural Network [37]

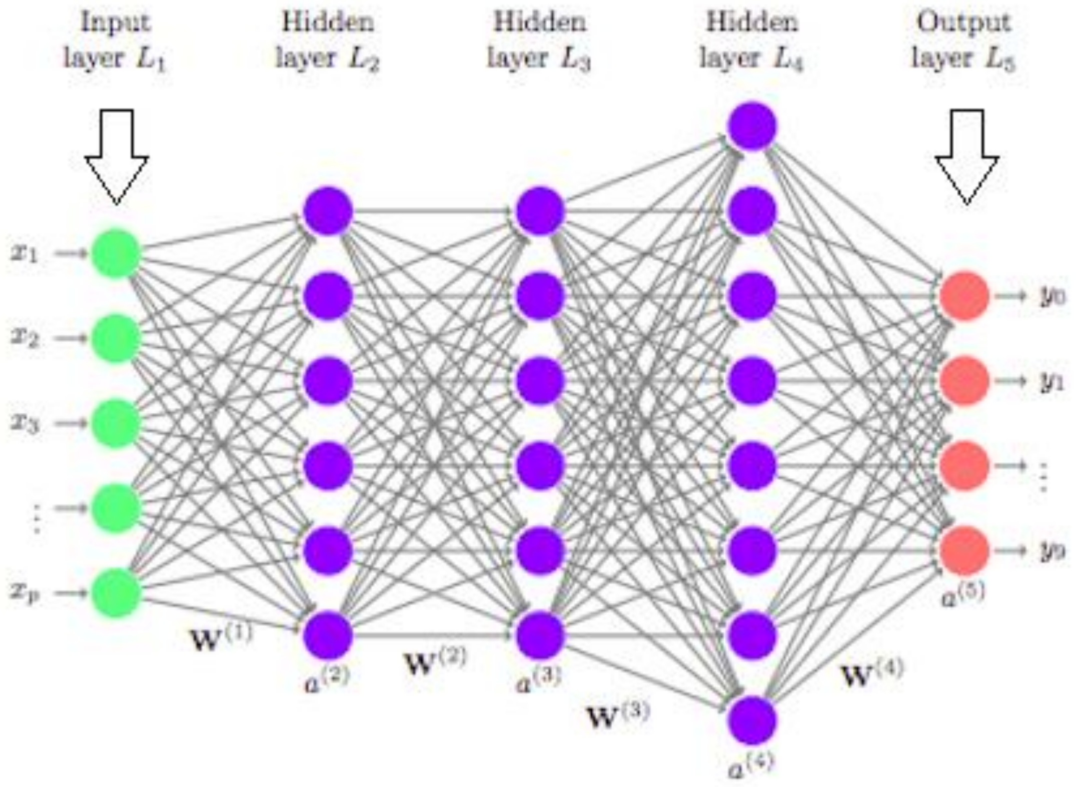


Image_07: Image showing the structure of a Deep Neural Network [43]

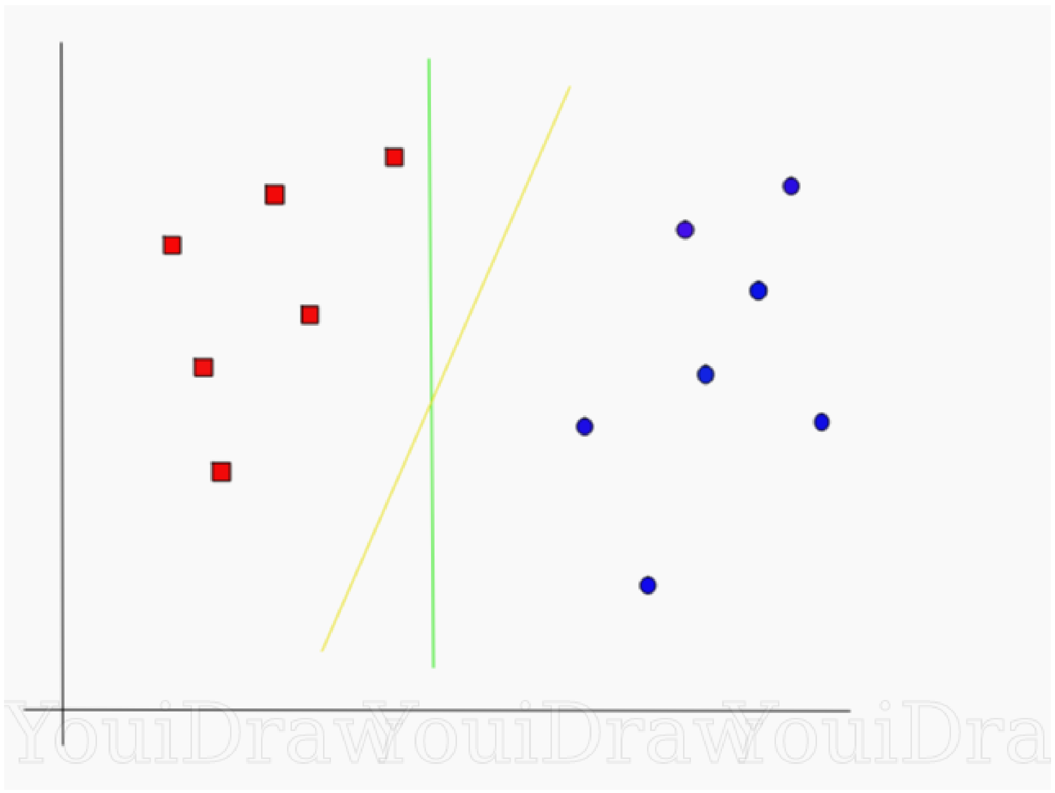


Image_08: A graph showing two classes in a Support Vector Machine (SVM) classifier [156].