



**DEVELOPING THE RECURRENT NEURAL NETWORK WITH LONG-SHORT  
TERM MEMORY AND WORD2VEC REPRESENTATION FOR SENTIMENT  
CLASSIFICATION**

**FALAH AMER ABDULAZEEZ  
ALKUBAISI**

**SEPTEMBER 2018**

**DEVELOPING THE RECURRENT NEURAL NETWORK WITH LONG-SHORT  
TERM MEMORY AND WORD2VEC REPRESENTATION FOR SENTIMENT  
CLASSIFICATION**

**A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED  
SCIENCES OF  
ÇANKAYA UNIVERSITY**

**BY  
FALAH AMER ABDULAZEEZ  
ALKUBAISI**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF  
MASTER OF SCIENCE  
IN  
COMPUTER ENGINEERING DEPARTMENT  
INFORMATION TECHNOLOGY PROGRAM**

**Title of Thesis: Developing the Recurrent Neural Network with Long-Short-Term Memory and Word2vec Representation for Sentiment Classification**

Submitted by **FALAH AMER ABDULAZEEZ ALKUBAISI**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.

  
\_\_\_\_\_  
Prof. Dr. Can ÇOGUN  
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

  
\_\_\_\_\_  
Prof. Dr. Erdoğan DOĞDU  
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully Adequate, in scope and quality, as a thesis for the degree of Master of Science.

  
\_\_\_\_\_  
Assist. Prof. Dr. Abdül Kadir GÖRÜR  
Supervisor

**Examination Date: 04.09.2018**

**Examining Committee Members**

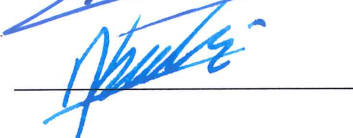
Assoc. Prof. Dr. H.Hakan MARAŞ (Çankaya Univ.)

Assist. Prof. Dr. Erdal ERDAL (Kırıkkale Univ.)

Assist. Prof. Dr. Abdül Kadir GÖRÜR (Çankaya Univ.)

  
\_\_\_\_\_

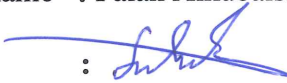
  
\_\_\_\_\_

  
\_\_\_\_\_

## STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and Presented in accordance with academic rules and ethical conduct. I also declare that, As required by these rules and conduct, I have fully cited and referenced all material And results that are not original to this work.

Name, Surname : Falah Alkubaisi

Signature : 

Date : 04/09/2018

## ABSTRACT

### **Developing a Recurrent Neural Network with Long-Short Term Memory and Word2vec Representation for Sentiment Classification**

**FALAH AMER ABDULAZEEZ ALKUBAISI**

**M.Sc., Computer Engineering Department**

**Information Technology Program**

**Supervisor: Assist. Prof. Dr. Abdül Kadir GÖRÜR**

SEPTEMBER 2018, 79 Pages

One of the major components of machine learning is classification. Sentiment analysis is one of the sub-fields of classification. It works on the methods that study and classify the opinions of people regarding their feelings and it extracts any underlying impressions toward subjects or even other texts. In this study, we worked on developing a neural network model for binary sentiment classification which can analyze data as being either positive or negative. Many papers conclude that probabilistic classifiers and linear classifier (SVM) methods are more accurate than Neural Network methods. In this study, we proved (demonstrated) that there is more space for development in the Neural Network methods field. We compared our results with four supervised methods: Naïve-Bayes, Maximum Entropy, Support Vector Machine and Stochastic Gradient Descent. We achieved better results than the results of the mentioned methods by using RNN (Recurrent Neural Network) with GLOVE (Global Vectors for Word Representation) and achieved a result of 91.04% accuracy.

**Keywords:** Sentiment Analysis, Deep Learning, Machine Learning, Movie Reviews, Neural Network.

## ÖZ

### **Developing a Recurrent Neural Network with Long-Short Term Memory and Word2vec Representation for Sentiment Classification**

FALAH AMER ABDULAZEEZ ALKUBAISI

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü, Bilgi Teknolojileri Programı

Danışman: Dr. Öğretim Üyesi Dr. Abdül Kadir GÖRÜR

Eylül 2018, 79 sayfa

Makine öğreniminin en önemli bileşenlerinden bir tanesi sınıflandırmadır. Duygu analizi, sınıflamanın alt alanlarından biridir. Duygu analizi insanların duygularıyla ilgili düşüncelerini araştıran ve sınıflandıran yöntemlerle çalışır ve konuya ve hatta diğer metinlere yönelik altta yatan izlenimleri çıkarır. Bu çalışmada, metinleri olumlu ya da olumsuz olarak analiz edebilen ikili duygu sınıflandırması için bir sinir ağı modeli geliştirmeye çalıştık. Pek çok makale olasılıksal sınıflandırıcıların ve doğrusal sınıflandırıcı (SVM) yöntemlerinin Yapay Sinir Ağı yöntemlerinden daha doğru olduğu sonucuna varmışlardır. Bu çalışmada, Sinir Ağ yöntemleri alanında gelişme için daha fazla alan olduğunu kanıtladık. Sonuçlarımızı dört denetimli öğrenme yöntemi ile karşılaştırdık: Naïve-Bayes, Maksimum Entropi, Destek Vektör Makinesi ve Stokastik Gradyan Descent. Bahsi geçen bu yöntemler ile karşılaştırıldığı durumda daha iyi sonuçlar elde ettik. RNN (Tekrarlayan Nöral Ağ) ile Glove (Kelime Temsili Global Vektörler) kullanarak % 91.04 doğruluk elde ettik.

**Anahtar Kelimeler:** Sentiment Analysis, Deep Learning, Machine Learning, Movie Reviews, Neural Network.

## **ACKNOWLEDGEMENTS**

My greatest gratitude and warmest thanks are due to my supervisor Assist. Prof. Dr. Abdül Kadir GÖRÜR of the computer Engineering Department at Çankaya University. His door was always open for me whenever I needed his help. His understanding, encouraging and personal guidance have provided a great basis for this thesis. His ideas and constructive notes have had a remarkable influence on carrying out this thesis. I am also highly grateful to my advisor Dr. Tolga PUSATLI for his help, advice and useful profound discussions.

I must not forget to record my deep gratitude to the members of my family for their endless patience and continuous support during the period of my study. Special thanks are due to my parents for their generous support, love and understanding. I owe a great debt of thanks to my brothers and sister who always encourage me to complete my study. Thanks are also due to my teachers and friends.

## TABLE OF CONTENTS

STATEMENT OF NON PLAGIARISM.....	iii
ABSTRACT.....	iv
ÖZ.....	v
ACKNOWLEDGEMENTS.....	vi
TABLE OF CONTENTS.....	vii
LIST OF FIGURES.....	x
LIST OF TABLE.....	xi
LIST OF ABBREVIATION.....	xii
CHAPTERS:	
<b>1. INTRODUCTION.....</b>	<b>1</b>
1.1 Aim of the study.....	2
1.2 Significant of the study.....	3
1.3 Research questions.....	3
1.4 Thesis structure.....	4
<b>2. Literature Review.....</b>	<b>5</b>
2.1 Introduction.....	5
2.2 Lexicon-Based Approach.....	8
2.2.1 Dictionary-Based Approach.....	8
2.2.2 Corpus-Based Approach.....	10
2.3 Machine Learning Approach .....	11
2.3.1 Unsupervised Machine Learning Approach.....	11
2.3.2 Supervised Machine Learning Approach.....	13
2.4 Conclusion.....	18
<b>3. Methodologies.....</b>	<b>19</b>
3.1 Introduction.....	19



3.2 Problem Statement.....	20
3.3 Methodology.....	20
3.3.1 Data Preprocessing and The Machine Learning methods Flowchart.....	20
3.4 Machine Learning Classification Methods.....	24
3.4.1 Naive Bayes Algorithm .....	24
3.4.2 The Maximum Entropy Algorithm.....	28
3.4.3 Stochastic Gradient Descent.....	29
3.4.4 Support Vector Machine.....	30
3.5 Deep Learning Method.....	31
3.5.1 Word to Vector (word2vec).....	31
3.5.2 Recurrent Neural Network.....	32
3.5.3 Layers in Deep Learning models.....	34
3.5.3.1 Embedding Layers.....	34
3.5.2.2 Dropout Layers.....	34
3.5.2.3 LSTM Layers.....	34
3.5.2.4 Activation Layers.....	35
3.6 Network Architecture.....	35
3.6.1 Stack of LSTMs.....	35
3.6.2 Bidirectional LSTM.....	36
3.7 Conclusion.....	37
<b>4. Design and Implementation.....</b>	<b>38</b>
4.1 Introduction.....	38
4.2 Tensorflow and Keras.....	38
4.3 The Machine Learning Methods.....	38
4.4 The Deep Learning Model.....	39
4.4.1 Word2vec Model.....	39
4.4.2 The RNN and LSTM Mechanisms.....	40
4.4.3 The Neural Network Flowchart.....	43
4.4.4 How It Works.....	45
4.4.5 The Neural Network Parameters.....	46
4.5 Conclusion.....	47

**5. Results and Discussion**.....48  
5.1 Introduction.....48  
5.2 The Machine Learning Methods Results.....48  
5.3 The Deep Learning Results.....55  
**6. CONCLUSION**.....61  
6.1 Conclusion.....61  
6.2 Future Work.....62  
REFERENCES.....63



## LIST OF FIGURES

<b>Figure 1.</b> Sentiment analysis classification methods.....	7
<b>Figure 2.</b> Naive Bayes classifier initializing steps.....	14
<b>Figure 3.</b> SVM support vectors points.....	15
<b>Figure 4.</b> CNN architecture.....	16
<b>Figure 5.</b> RNN sequential architecture.....	18
<b>Figure 6.</b> Preprocessing steps of the dataset .....	21
<b>Figure 7.</b> Naive Bayes Algorithm.....	27
<b>Figure 8.</b> The estimation model for naive Bayes classifier for sentiment Analysis.....	28
<b>Figure 9.</b> The SGD Algorithm.....	30
<b>Figure 10.</b> The Mikolov Word2Vec architecture.....	32
<b>Figure 11.</b> Recurrent Neural Network.....	33
<b>Figure 12.</b> The RNN with LSTM model.....	33
<b>Figure 13.</b> Stack of two layers LSTMs .....	36
<b>Figure 14.</b> The Bidirectional LSTM Architecture .....	37
<b>Figure 15.A</b> Glove vs CBOW .....	40
<b>Figure 15.B</b> Glove vs Skip-gram .....	40
<b>Figure 16.</b> The time steps inputs feeding in RNN .....	41
<b>Figure 17.</b> Sequential process in RNN.....	41
<b>Figure 18.</b> The LSTM unit architecture.....	42
<b>Figure 19.</b> Sentiment analysis for sentence.....	43
<b>Figure 20.</b> Building the tensor step in TensorFlow.....	44
<b>Figure 21.</b> The sentiment analysis flowchart.....	44

## LIST OF TABLES

<b>Table 1:</b> The dataset statistics.....	21
<b>Table 2:</b> Count vectors for the three documents.....	23
<b>Table 3:</b> Sentiment analysis works comparison.....	49
<b>Table 4:</b> The results of the methods used in [35].....	50
<b>Table 5:</b> The comparison of the methods used on IMDB sentiment analysis.....	53
<b>Table 6:</b> The results of our model.....	60

## LIST OF ABBREVIATIONS

NN	Neural Networks
NB	Naïve Bayes Classifier
ME	Maximum Entropy
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
w2v	Word2Vector
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
LSTM	Long-Short Term Memory
GLOVE	Global Vectors for Word Representation
TF-IDF	Term Frequency- Inverse Document Frequency

## CHAPTER ONE

### Introduction

A human can easily describe his opinions and make eloquent demands and the main underlying purpose of machine learning and Artificial Intelligence is to have the machine act like a human being. In natural language processing, there are many methods that can be implemented with linguistics, and from those fields, we have sentiment analysis (SA). The purpose of SA is to analyze user sentiment from a text.

According to [1], with the increase in social media and social media networks, the size and amount of information has increased, and daily there are millions of reviews and much information from users. In order to make that information useful, sentiment analysis has become one of the modern sciences for processing such data. Sentiment analysis is good for understanding the sentiment in a text from a real user and to check whether it is positive or negative. It can be used to check the popularity of products, users, subjects and so on. Recently, the collection of user data has become easier than before with a return to a growth in technology, especially social media, with which the user can share his opinion about products or anything else by publishing comments. This development is very useful for retrieving data for analysis.

Another source for data retrieval is blogs, which contain many reviews and opinions for many users on a daily basis. They (blogs) are considered a rich source for the acquisition of data for sentiment analysis. Sentiment analysis is one of the modern areas of text classification which is still new and has further room for development. Initially, scientists depended on the big data collected by users and the focus was on only those kinds of data. After that, the focus moved to the usage of social media and the activities of users on those websites, known as microblogs. There are many microblogs on the Internet, including the *Twitter* platform. As can be inferred from its name, the primary

part of it is the so-called “tweet,” which is a comment by a user consisting of a maximum 140-character message.

*Twitter* is a social media platform which consists of millions of user reviews the numbers of whose retrievals exceed the number of users of reviews and comments on a daily basis. Tweets may contain a user’s opinions and his expressions about a specific subject and even a tweet from another user. However, there are many difficulties such reviews with many obstacles and issues including spelling, abbreviations and shortcuts being used widely to express the feeling(s) of the user known as “emoticons.” Moreover, a tweet might be very short and composed of a few words. All these complexities require a powerful method to handle them or replace the tweet with a more general dataset that is clear and rich in words per review.

Many techniques use in term of sentiment analysis. There are lexicon based methods which depend on the linguistic distribution of words in a text, known as “semantic orientation,” to classify a text as being either positive or negative [1] [2] [3] [4] [5]. Other methods include machine learning methods which train modules with pre-classified datasets to define the sentiments of text as well as other datasets for testing [1] [2] [6] [7] [8].

### **1.1 Aim of the Study**

In this thesis, we did not use the dataset from the *Twitter* platform which returns all the complexity in *Twitter* comments and reviews, such as slang, irony, misspellings and so on. The dataset we are using in our work, which a binary sentiment classified dataset, is a number of IMDB movie reviews from Stanford researcher Andrew L. Mass [9]. Many machine learning approaches have been covered with their perspective results on the same dataset and compared with our proposed method results. In general, the most used algorithms in sentiment analysis are the supervised learning methods, such as Naive-Bayes and similar methods. Our work covers the supervised learning methods and compares between probabilistic classifier and linear classifier (SVM) results with Neural Network model results. We will focus on the Recurrent Neural Network (RNN) as the main part of our work.

## **1.2 Significance of the Study**

In this thesis, we search for efficient algorithms for the sentiment analysis task and discover the most accurate among the current methods being used in the research area. We investigate the classification algorithms in machine learning and measure the accuracy of each method with a dataset. Later, we provide a comparison of the results of the methods. Finally, we compare those results with the results gained from our model and show the accuracy of our RNN. Every test was performed with the same dataset. An advantage of this dataset was that both the training and testing data were labeled as binary classifications, the label being either positive or negative. The algorithms that were used as supervised methods were the Naïve-Bayes algorithm, Maximum Entropy, Support Vector Machine and the Stochastic Gradient Descent. All the algorithms in the supervised methods had many versions of the n-gram classifier. This classifier is unified as unigram, bigram and trigram. Additionally, we used a composed classifier from the unified parameters.

## **1.3 Research Questions**

- Since most of the work on sentiment analysis focuses on probabilistic classifiers and SVM as the main methods, can the neural network methods provide a better Result than those methods?
- Do pre-trained vectors provide better results than vectors generated Through training?



## 1.4 Thesis Structure

There are four further chapters in addition to this first chapter:

**Chapter 2** presents the literature review of the thesis and it will cover every method used in the sentiment analysis task from a general perspective. Moreover, it will cover all the challenges in the field of the sentiment analysis and a detailed explanation of lexicon-based methods.

**Chapter 3** presents the details of the methods used in the sentiment analysis. It will cover the implementation of those methods in machine learning and an overview of the classification techniques. The dataset used for training and testing is also covered in this chapter.

**Chapter 4** presents explanations of the mechanisms of the methods and tools used in the work. It covers ideas used in previous work and the contribution of the new study. This chapter also presents information about the sequence of data flow during training and testing in the practical phase.

**Chapter 5** presents the experiments of the methods mentioned above and the results of those methods in addition to other results from testing those methods with a new dataset that was not used for the training. After comparing the results of those tests, we can define the best method among them for sentiment classification.

**Chapter 6** presents the conclusion of the work and future work.

## **CHAPTER TWO**

### **Literature Review**

This chapter presents an overview of all the methods that can be used in practice for sentiment analysis and all general information about the algorithms covered by scientists in the field.

#### **2.1 Introduction**

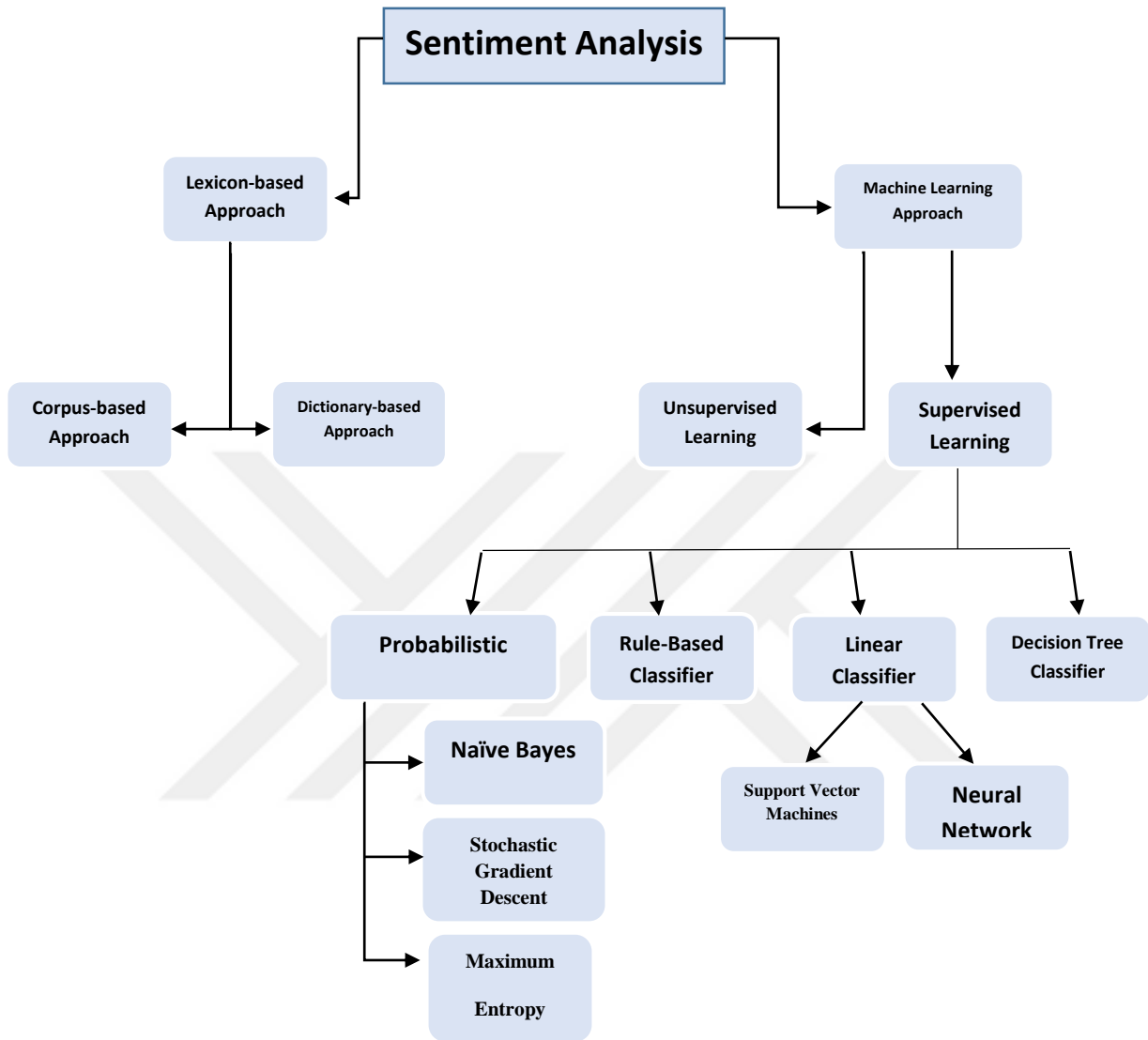
The work in sentiment analysis started in the 1990s and it has not drawn too much attention to scientists until the beginning of the new century due to its significant effects in many areas in computer science and other fields [1]. Data started increasing and many researchers started collecting data and labeling the data with opinions for those data. This development attracted the attention of scientists and pushed them forward into the field of sentiment analysis. Bing Liu concluded that sentiment analysis was a mining of user opinions and analysis of those opinions associated with other entities, such as organizations, persons, suggestions, products, politics and many others. The data that contain opinion are processed using sentiment analysis, after which they are classified with a sentiment. Most sentiment analysis tasks categorize texts in a binary manner as being either positive or negative [10] [11]. Other types of sentiment can be classified into more than two segments [12] [13] [14], and those categories may be positive, highly positive, negative, highly negative, natural, or they can even be emotions such as happiness, sadness, joy, fear, etc.

Some of the most compelling reasons that bring sentiment analysis to the forefront are the decisions that can be made depending on sentiment analysis. Organizations focus on sentiment analysis to extract opinions from reviews and feedback from people. This procedure mitigates the effort exerted by those organizations to collect feedback about

particular products or subjects. The Internet, especially the social media, has helped to provide such organizations with information. Furthermore, this extracted information helps to reduce the cost of creating surveys to collect data about products or services. Instead, it can focus on any data that can be gathered from the Web at no cost and it saves any time consumed on surveys. Although information can be freely gathered, the data can accrue some noise or unwanted data which can affect the task, which will necessitate some preprocessing to use it in sentiment analysis. For this kind of problem, sentiment analysis has many methods to overcome this problem easily by processing on many levels [1]:

- **Document level:** At this level, opinions can be gathered from an entire document which should be related to one topic [6].
- **Sentence level:** At this level, each sentence in the document can be dealt with as a small document from the primary document. Then, the sub-document is classified as an objective document and ignored, or it is classified as a subjective document and the sentiment is extracted from it [11].
- **Aspect level:** The sentiment can be extracted from the aspects of an item. Somehow it is considered to be a very general method but still effective in terms of sentiment analysis [15].

Sentiment analysis classification methods can be categorized into two categories, namely lexicon-based methods and machine learning methods [2]. For a clear review about these two categories and their related sub-categories, see Figure 1.



**Figure 1:** Sentiment analysis classification methods [2]

## **2.2 Lexicon-Based Approach**

The lexicon-based approach is one of the main components of sentiment analysis. This approach uses a lexicon of terms in a document and each is has a value associated with it called a score. A term can be related to a word from a language or even a phrase [16]. The sentiment calculated depends on the presentation value of the score from the lexicon. There are two methods in the lexicon-based approach: the dictionary-based approach and the corpus-based approach. These two methods will be discussed in the following paragraphs.

### **2.2.1 Dictionary-Based Approach**

The main part in this method is the lexical database which contains the score of the words and those scores measured in terms of the sentiment of the words. The sentiment is extracted from a document depending on the score of the words from the lexicon database. Seed words should be the initial set for the task and should not be very large (a maximum of 30 words) [4] and those words should have high polarity, such as the words “good” or “bad” [1] [3]. A polar word should be Fed next to augment the set of words and check for any synonyms in the database. There are many types of lexicon database, some of which are free and others commercial. As examples of this type of database, there is SentiWordNet [17], WordNet [18], SenticNet [17], HowNet [18] and so on. This comparative procedure is performed here iteratively, which means that the set of words will be updated on each iteration and the size of the set will also be expanded. The search will continue until there no words remain to be added to the set. A study was published by Hu and Liu in 2010 which focused on gathering customer reviews about a feature of a product that has sentiment. Reviews are gathered as a summary. As an example of this process, a product may be a camera and the sentiment features may be the quality of the photographs taken or the size of the product and depending on the data collected from those features, the reviews will be grouped into positive reviews and negative reviews about the camera. The procedure is performed particularly by collecting polar words from a sentence, such as the adjectives. The score is retrieved for the word or its synonyms and even depending on the antonyms opposing the polarity of the antonyms. By summing the scoring of the polar words, the sentiment can be predicted for the sentence. The results of work in [4] present good results and an average accuracy of 84%. Another study performed by Kim and Hovy followed several

methods classifying sentiment analyses using the topic of the text. There were several levels for classifying texts. Initially, a classifier was used to find the polarity of each word in the text. The second classifier provided the polarity for the entire text regarding user opinion. In this paper, the authors used a short list as seeding for the initial stage similarly to [4]. By searching for the synonyms and antonyms in the WordNet lexicon database in this paper, the list would expand with new feeding words. However, the authors proved that a number of synonyms yielded the opposite polarity from the original word and in some cases, it yielded a natural polarity which affected the final result of the sentiment and affected its accuracy. Moreover, the authors indicated that the polarity of the words should be measured more accurately by providing a wide range of numbers of polarities to distinguish the strength of the positiveness and negativeness. By performing this procedure, the number of ambiguous words of ambiguity will be reduced and accuracy will increase. The work provides four parts in a sentence that can provide the sentiment of that which is similar to the person holding the opinion of that sentence. Moreover, there are three models for resolving orientation in sentences. The first one is the elimination of a negative when another negative appears. The second and third models iterate the words of the sentence and calculate the strength of the sentiment with a harmonic mean in the second model and a geometric mean in the third model. As a result, the authors proved that the best results would be collected from the first model and in the region where the user determined the sentiment forwarding to the end of the text [19].

Another study was performed by Park and Kim in 2016 [20] on *Twitter* data. The work concluded by gathering seed words and searched for synonyms/antonyms from three different dictionaries. The collected words were used to increase the lexicon of the words; then those data were used to classify the data. This method performed better than the traditional dictionary-based method. However, there were a number of problems in this method, the first of which was the time consumed to collect the synonyms and antonyms. The next problem was the informal words used in tweets, which causes difficulties retrieving their polarity. In conclusion, the dictionary-based approach had not provided the best results for the sentiment analysis due to the distribution of words and their polarities [2].

### 2.2.2 Corpus-Based Approach

The corpus-based approach can be adjusted in two stages according to Bing Liu's opinions [1]. The first stage is the identification of the polarity of the words in a sentence using an initial list of words that have a strength polarity. The second stage is the building of a lexicon database from another database for the words from the sentence. He concluded that the same word might receive the opposite orientation in the text depending on the complete distribution of the words in a text. Hazivassiloglou and McKeown [5] focused on extracting the semantic orientation from the compound adjectives in a text and used the same initial seed concept. They used a special rule to extract the polarities from words. The compound adjectives gave the same polarity when they were compounded with the stop word "and." In another case, the word "but" was used to produce the opposite polarity for compound adjectives and the same applied for "neither-nor" and "either-or" and the stop word "or." In many cases, those rules did not produce accurate results, so the researchers compared the polarity of the compound adjectives with the polarity of the words separated from each other using the linear-regression model. As a result of this procedure, the predictions were presented in a graph and clustered into two parts, one for the negative and the other for the positive. Finally, they produced an accuracy of 90% in their work.

Since words in sentiment analysis have underlying orientations as semantic, they are distributed depending on the corpus. A method was developed by Ding [21] to determine semantic orientation. He suggested that some linked adjectives have some dependencies, such as those words that compound with "long," "short," etc., and depending on this representation, the polarity can change. Ding covered in his work many objects in a lexicon, such as phrases, idioms, and words. He also extended the list created in [4] by including nouns and adverbs. Moreover, he added more than 1000 idioms with clear sentiment polarities. After preparing the modified lexicon database, the polarity of each word was calculated depending on the rules defined by the authors. The polarity was summed with the score function to give the total score of the sentence. The function provides better results than the normal summing in the previous work [4]. The authors proposed three methods to overcome the context dependent words, including words used with "but." These methods are used to classify the connectivity to deal with them [21]:

- Intra-sentence conjunction technique
- Pseudo intra-sentence conjunction technique
- Inter-sentence conjunction technique

The authors concluded that with the help of these methods, the result of the sentiment is better than those of the previous methods.

Because of the limitation of the words used in the corpus-based method, the lonely corpus-based method shows lower accuracy than the dictionary-based method. However, it is still better in terms of rebuilding lexicon databases. Another weakness in the corpus-based method is the time consumed on processing due to the number of words used to expand the lexicons, and with increases in it, performance drops [22].

## **2.3 Machine Learning Approach**

Machine learning approaches in sentiment analysis may be classified as supervised machine learning methods and unsupervised machine learning methods. These two types are explained below.

### **2.3.1 Unsupervised Machine Learning Methods**

All of the datasets that are not categorized with labels are unsupervised data and can be grouped into the unsupervised machine learning approach. Those methods examine the data and classify them according to their pattern. The data with similar features will be identified together under a category. It is easy to collect unsupervised data by retrieving them from any resource.

In Turney's paper [6], unsupervised machine learning methods are used to classify reviews for sentiment analysis. Each review has two features either recommended or not recommended and the review should have one of these features. Using the tag impression, Turney collected the phrases that consisted of two words. The tag was designed in such manner to contain reviews that had high sentiments associated with the tag subject. As an average, each review contained 5 patterns of phrases. Moreover, a Part-of-Speech (POS) tagger was used in the work to filter the phrases that were required to be collected for the task. Occasionally, the retrieved phrases from the



reviews had to be calculated according to their semantic orientation. Moreover, the work had an information retrieval (IR) algorithm and a Pointwise Mutual Information (PMI) algorithm to find the semantic orientation by calibrating the semantic similarity for the words (between every two words). Any phrase meeting the condition would be considered the first word as the pattern and the next one as a reference. The references would vary; for example, the word “weak” would be considered as one-star due to its low polarity in comparison to the high-polarity word “strong.” In the review, the grading system would start from one up to five stars. The PMI would calculate the difference in the semantic orientation for every two phrases, in our case, the PMI of the phrase, “weak” and the phrase, “strong.” In this scenario, the semantic orientation would be positive with any word coming with “strong” as a reference and negative for any word associated with “weak.” The PMI can be calculated by building a co-occurrence table for the words, which means that for every two words appearing together, they are considered a hit and added to the table. This is performed with the AltaVista search engine, which is used for this procedure in the paper. Finally, the sentiment is calculated for the entire review and will be either recommended (positive) or not recommended (negative). A 74% accuracy was achieved in the work.

Another work [23] published by Rothfels used the unsupervised machine learning approach for sentiment analysis for movie reviews. The work was inspired by [24], which was used to classify text in the Chinese language. Initially, positive seed words should be collected from the review. A list of seed words is collected by iterating the classification of those words in the document (review). The text of the document is segmented into pieces and each piece prepended with punctuation marks at the beginning and appended with punctuation marks at the end. The text with a domination of either positive or negative zones helps to predict the sentiment of the text. In practice, a text that has more positive zones than negative zones is considered to be a positive document and vice versa. Rothfels attempted to use different types of seed list by expanding it. He used many types of n-grams, including the bigram, trigram and 4-gram, but only the 4-gram provided an acceptable result relative to the first two types. With a regrettable result, the work continued with a second attempt. The author started with highly semantic words as a seeding. However, even with this modification, the result was not sufficient and the same results occurred when they used the K-means clustering algorithm for scoring. Finally, he used a method from [6] to measure the

semantic orientation of phrases by collecting a two-seed word list, one for the positive and the second for the negative. The semantic orientation was measured again. The new list of words consisted of highly semantic words only. Later, by iterating the semantic orientation on the words, the accuracy increased by 15%.

### 2.3.2 Supervised Machine Learning Approach

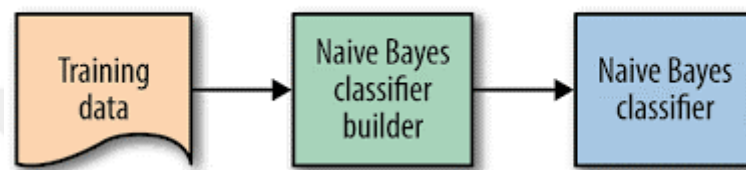
Any data with labels are supervised data. This kind of data is used to train the algorithms, which is a common feature in almost all the supervised machine learning methods. After the training step, the output is compared with the input dataset. The predication is calculated by comparing the predicted value with the labeled value. There are many representations for a document, but the most well-known is the bag-of-words, wherein the document is represented as a vector of the unique terms in that document  $d = (w_1, w_2, \dots, w_i, \dots, w_N)$ , where N is the total number of unique words in a dataset and  $w_i$  is the weight of the term [7]. The training data should be represented as a vector of features and this occurs by creating a list with the same value as N from the training data. There must be a feature model to perform this procedure. Many feature models are represented, including [7]:

- **The Binary Model:** This model represents the weight as 1 if a term appears in a document of 0 for the opposite.
- **Term Frequency (TF):** The number of appearances of a term in a document.
- **TF-IDF (Term Frequency-Inverse Document Frequency):** The TF here recognizes every term with the same importance, whereas the IDF represents the importance of a term in the documents.
- **Information Gain (IG):** This measures the popularity of a feature in a class in comparison with others.
- **Chi-Square Test:** This is explained in the following chapter.

There are many methods used for sentiment analysis; however, we have covered the most common algorithms in the field of sentiment analysis. The first method is the Naïve-Bayes algorithm [7] [25] [26] [27] (see Figure 2 to understand the initializing of the Naïve-Bayes classifier). This algorithm is used to calculate the assumption of

features that are independent from the Bayes theory. It shows the probability of a document being related to a class. The algorithm is easy to understand and can be implemented without complexity and it is fast with a high learning rate. The results vary from one test to another, but usually it performs with high results in comparison with many other algorithms [11] [26] [27]. The only problem here is that when there is a real-world problem, because most of the features are dependent, it conflicts with the naive theory.

### Building a Naive Bayes classifier



**Figure 2:** Naive-Bayes classifier initializing steps

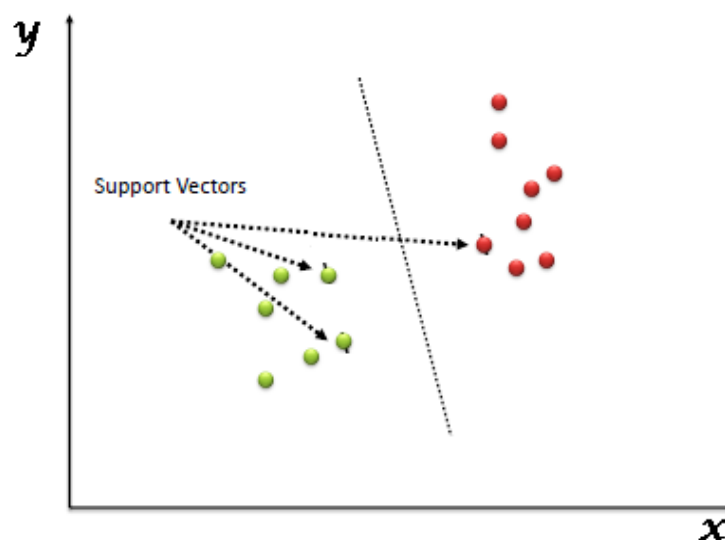
Another algorithm is the Maximum Entropy model, which states that the probability of a document belonging to a class should be estimated at the maximum value [25] [26]. The formula for the maximum entropy is:

$$P(c|d, \lambda) = \frac{\exp[\sum_i \lambda_i f_i(c, d)]}{\sum_{c'} \exp[\sum_i \lambda_i f_i(c', d)]}$$

Where  $c$  is the class,  $d$  the document, and  $\lambda$  the weight of the classification indicator  $f_i$ . Since the ME does not need to take in consideration the feature independence, it might give a better result than the Naïve-Bayes at least theoretically. Meanwhile, the learning is slower than the Naïve-Bayes as well as its implementation.

The next method is the rule-based classification algorithm. This approach contains a set of rules created by professionals by analyzing the features in a specific domain, and depending on the usage of many rules, the results are promising; however, it still takes much time to process the data. This algorithm was created by Chikersal in 2015 [28] depends on data gathered from *Twitter*. He used the rules that take in action of the appearance of high sentiment terms in a tweet with emoticons. In his paper, the author later used the support vector machine classifier (SVM) to classify the removed emoticons deleted from the dataset for training as well as an L1-regularization and

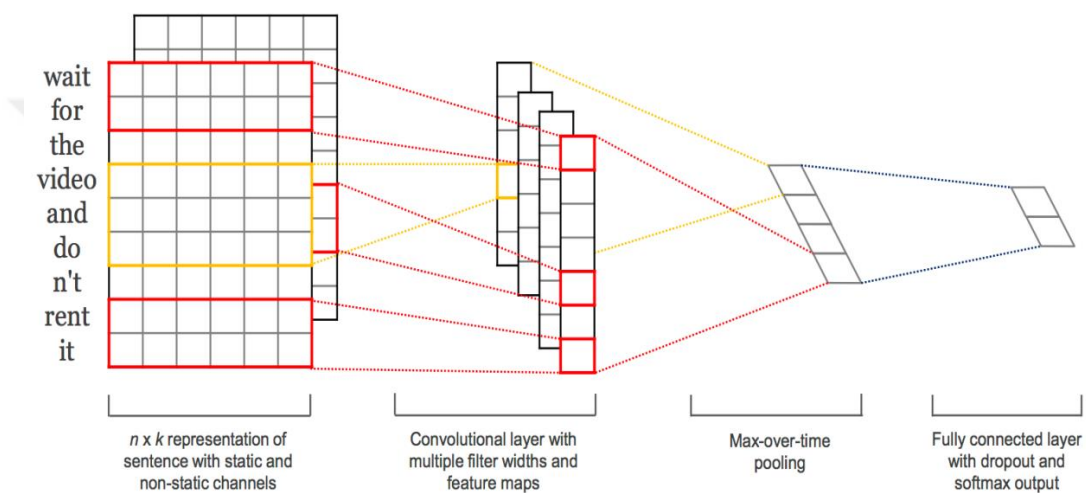
linear-kernel. The work also covered n-grams and POS tags and various types of lexicon database, such as SentiWordNet, the Bing Liu Lexicon and the Sentiment140 lexicon as well as many others. The application of all these methods was to combine more than one method into one composite method to enhance recall and precision. First, tweets were labeled as natural by the SVM and then analyzed with the rule-based approach to represent the label for the data. Later, all the results were totaled showing a good result. Many other papers also used SVM [25] [26] [27]. The entire algorithm could be concluded to be data distributed as points in space and each point associated with an appropriate class and those points would need to be separated and joined with the points related to the same class with a hyperplane. The training step here was to determine how best to separate the data so that the distances between the nearest points of each class to the hyperplane should be the same. Those points were called the support vector points (Figure 3) and any changes in those points or removing them would change the position of the hyperplane [29]. In many papers, the SVM brings better results than other algorithms, such as the Maximum Entropy and the Naive-Bayes [27], but it also has a time complexity which makes it a bad choice for classifying large datasets.



**Figure 3:** SVM support vector points

The modern common development in recent times is the Artificial Neural Networks or Neural Networks (NN). Similar to the actual human neural network, the neural network performs similarly to human neural networks to solve problems by performing a great number of calculations to achieve a satisfactory result. Simply, it is a network of

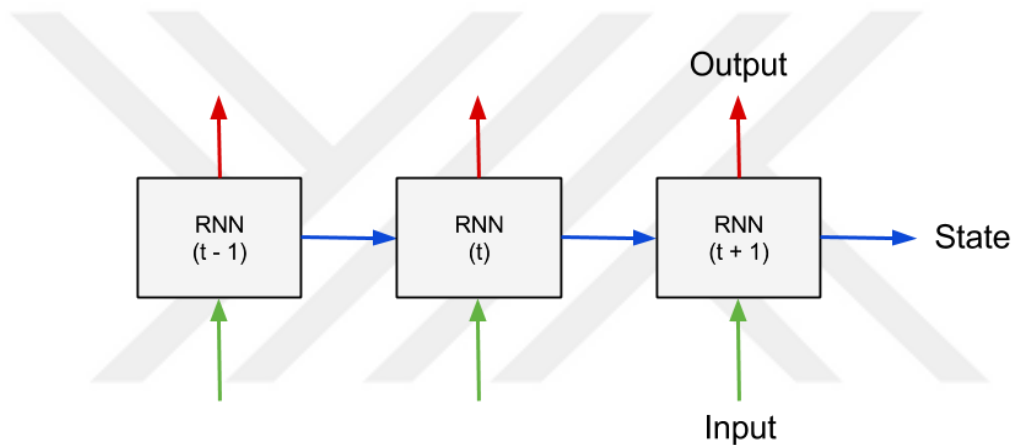
interconnected neurons distributed in many layers and it learns by modifying the weights of the neurons and repeating operations until the network learns the best value and becomes satisfied to answer the request for which it was built. There are many types of these used in classifying texts, and the most common types are the Convolutional Neural Network (CNN) and the Recurrent Neural Network (RNN). The CNN consists of interleaving layers with convolutions, sub-sampling and fully connected layers. It works in random order to cover every case of classification (see Figure 4) for the CNN architecture.



**Figure 4:** CNN architecture

Severyn and Moschitti worked on sentiment analysis with the deep Convolutional Neural Network. A network with one layer as a convolutional layer was used with a Rectified Linear Unit (ReLU) non-linear Activation function and max pooling layer followed by Softmax layer for classification [30]. They used word embedding as the weights for the input, which was developed by Mikolov [31] and which is a neural language model for word representation. The model performed well with data gathered from *Twitter*. In another work [8] performed by Kim, the CNN was used for text classification. A model was proposed to classify phrases into positive and negative as well as fine-grained classes. The model also classified the questions of 6 groups and classified sentences into subjective and objective. Many datasets were used in the research. As in the previous work [30], Kim used the word embedding model from [31]. The network consisted of convolutional and max over time pooling layers as well as fully connected layers. The model performed very well and the use of pre-trained word

embedding showed a very good result for feature extraction. Another high-performance architecture of CNN similar to the Kim model presented in [32] is the RNN network, which has the ability to feedback through neurons, thereby helping in storing the data from previous steps for use in following steps for more accurate calculations (see Figure 5 for the RNN model). After this step, the output would be tested with a test dataset and the learning rate would be determined by the error rate to give more accurate values. The benefits of using the RNN is that it is sequential, which makes it good for text data and predicting the next word in a sentence [34]. Capturing the context from the previous steps and using it to predict the next step made the RNN good at dealing with sentences.



**Figure 5:** RNN sequential architecture

Liu [33] used an RNN to classify text with multi-task learning. The first task of classification was binary-level. Another task classified sentences into subjective and objective, which performed binary classifications at the document level. In his work, the author built three different architectures to share information and feed it into the text sequence model. The first architecture had a shared layer for all the tasks above. The second one used different layers for different tasks. The third had a shared layer for the tasks but it took one task at one time at a specific level. The author claims that in some tasks, they outperformed the state-of-the-art methods. The last classifier in the supervised approaches is the Decision Tree [34]. It used a hierarchical architecture to classify texts. There are two types of nodes in a decision tree, namely the leaf node which has the value (label) of the word, and the decision node, which contains the

output of the attribute to make decisions. All the steps to move in the tree occurred recursively.

## **2.4 Conclusion**

In this chapter, we presented general details about the approaches and their algorithms that have been adapted in the text classification and sentiment analysis field. All the experiments mentioned in the chapter had been tested by the authors for the task, and among all the approaches mentioned above, most conceded that the machine learning algorithms showed the best results. The supervised machine learning methods were preferred due to their good results in comparison to the other supervised techniques [27]. Despite the simple implementation and concept of the Naïve-Bayes algorithm, it showed the best results with the neural network models in sentiment analysis tasks. Many papers show reasonable results for neural networks. As for the natural language processing (NLP) science, they return to the ability of those models in mitigating the training steps to learn the optimal structure for the model as well as their comparison quality. The models showed an exceptional classification in comparing the old methods. Finally, due to all of these reasons, the Naïve-Bayes and similar methods with Artificial Neural Networks were selected for sentiment analysis of movie reviews.

## CHAPTER THREE

### Methodologies

#### 3.1 Introduction

Due to modern developments in technology, the computer sciences received the most space of this development, which returns to the overlapping of information technologies in other fields. Nowadays, people use the Web prolifically in almost every aspect of life. During the day, they share their activities and their opinions on subjects and products, such as on *Amazon* and other shopping pages. Many of those reviews stay active for some time, and by using those reviews, the new customer can check those reviews to seek opinions about products as to whether they will purchase them. Another website for sharing reviews is *e-booking*, which is a website for checking hotel and flight availability and reservations. In general, it is very useful for traveling. Moreover, *Twitter* is also a good site for users to share their opinions and feeling about their lives and products, and even political issues. However, the problem of *Twitter* is the incidences of misspelling and grammar mistakes because of the modern style of chatting and the differences between communities and languages, which necessitates more accurate and understandable manners of expression that do not overlap with English grammar. This type of style can be found on many websites that discuss a specific subject or field. The problem with this kind of website is that such websites lack communities, which in turn produces reviews that are too few in number or have an unexpected quality. Nevertheless, there are websites with reviews that are rich in content of reviews, such as movie review websites and blogs. The most well-known type of these websites is the IMDB and *Rotten Tomatoes* website. The sentiment analysis and text classification might not have received much attention until recent years, but it is not a new area. Many researchers depend on the products receiving short



reviews at the beginning in this field [4] [6] [21]. Moreover, many others use movie reviews due to the features mentioned above and the quality of the reviews [6] [23].

### **3.2 Problem Statement**

The main focus in this work is movie review sentiment analysis which is performed by distinguishing positive and negative reviews. The work covers the machine learning methods that have been implemented in [35] and our part of the implementation covering the deep learning part. Moreover, the work covers the methods we presented in the second chapter. The Naïve-Bayes algorithm, Maximum Entropy, Support Vector Machine and the Stochastic Gradient Descent are the most popular methods in sentiment analysis, so we covered those methods and compared them with the neural network. Prior to commencement of the training, we used the weights for the model, particularly the word2vec model, which are pre-trained weights.

### **3.3 Methodology**

In this part of the chapter, all the underlying concepts and algorithms for the sentiment analysis task are explained in detail and additional information about the dataset used in this study is presented.

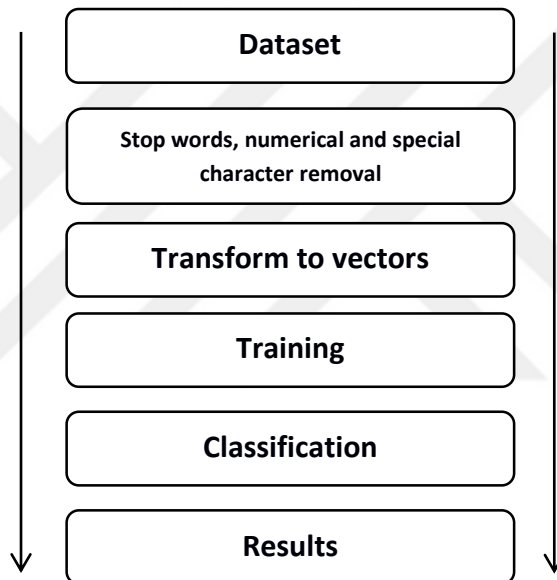
#### **3.3.1 Data Preprocessing and the Machine Learning Methods Flowchart**

In this work, a dataset for IMDB (Internet Movie Database) was used for training and testing the models [9]. The dataset contains 25,000 movie reviews and all are binarily labeled (positive, negative). Half of them are labeled as negative and the other half as positive as the training data. Furthermore, 25,000 more movie reviews are labeled in the same manner to test the model with 12,500 being positive and similarly 12,500 being negative. The dataset contains an unsupervised dataset without labels for bag-of-words tasks if needed. The training part and the testing part are the only parts needed in the task. Table 1 presents the statistics of the dataset.

**Table 1:** Dataset Statistics

Name	Part	Positive	Negative	Total
IMDB Reviews	Training	12,500	12,500	25,000
	Testing	12,500	12,500	25,000

The dataset has been processed on many levels to remove any noise and stop words. After removing the stop words from the dataset, it is vectorized and injected into the machine learning algorithm to start training, followed by classification and finally the results (see Figure 6).



**Figure 6:** Preprocessing steps of the dataset

The preprocessing step is performed according to levels and each level processes text in a different manner until achieving the final result.

**Step 1:** The IMDB dataset is taken into consideration by checking and saving the labels of the review to have them be references for the following steps.

**Step 2:** We remove the noise and unwanted data, such as:

1. Stop words: These are words that do not have a high effect and are the most frequent words in English which can be repeated many times in a review. Examples include “the,” “that,” “is,” “are,” etc.

2. Numeric characters: All the numbers with any type of numerical representation.
3. Special characters: Characters that are not alphabetical or numerical and which do not have any semantic effect and can produce ambiguity in the classification, such as \$, #, %, &, etc.

**Step 3:** In this step, the review should be converted into a numerical vector. For this operation, there are many methods for the conversion that are to be considered:

- **CountVectorizing:** This occurs by tokening the words and calculating the counts of each word. In particular, it is a mix of tokens and counting operations. As a result of this mixture, a matrix is generated from this process called a “sparse matrix.” To have a clear understanding for the process, the following example is provided:

For the CountVectorizing process, [36] provides a good example of the operation. In the example, there are three different documents and each document has sentences different from the others, as shown below:

Document 1: Movie is nice

Document 2: Movie is bad

Document 3: Movie is fine

The sparse matrix will be as shown below in Table 2, which shows the number of documents and builds a vector representing whether a word appears in the sentence. Hence, the terms might be higher in number than the number of words in the documents completely depending on the terms from all the documents to create a general vector representation. In the example, the number “1” means that the word appears in the document and opposite for the “0”.

**Table 2:** Count Vectors for the Three Documents

	Movie	Is	Nice	Bad	Fine
Document 1	1	1	1	0	0
Document 2	1	1	0	1	0
Document 3	1	1	0	0	1

- **TF-IDF:** This method is well known in the information retrieval and text mining fields. It simply shows the importance of a word in a document and for the entire corpus (all the documents in a data file). TF (Term Frequency) refers to the number of occurrences of a word in a document. IDF (Inverse Document Frequency) shows the importance of a word in whole documents (corpus) [36]. To calculate the TF-IDF value, the example below provides a clear understanding of the calculations.

If there is a document which consists of 1000 words and the word “awesome” appears 20 times in that document, then the TF of the word “awesome” is  $20/1000 = 0.02$ . For the IDF of the document, there is a need to know the total numbers of words in the corpus. If we have 1 million words in the corpus, then the IDF of the document will be  $\log(1,000,000 / 1000) = 3$ . Finally, the TF-IDF of the word “awesome” will be  $0.02 \times 3 = 0.06$ .

**Step 4:** After vectorizing, the vectors become inputs for the supervised machine learning algorithms. In this dissertation, there are four different algorithms for the sentiment analysis task. Every algorithm is explained in the following paragraphs separately:

- **Naïve-Bayes:** This algorithm classifies an event depending on the probability of its occurrence and learns the patterns for the events. It classifies many documents after learning the patterns [37].
- **Maximum Entropy:** By categorizing the data with groups within the maximum distribution by considering that there is independent between the data. That distribution is used to explicit the training data and use them again for testing [38].

- **Stochastic Gradient Descent:** The SGD algorithm is good when the data are large. It iterates repeatedly to estimate the gradient of a randomly selected part from the data [39].
- **Support Vector Machine:** Using the hyperplanes, the data are analyzed and grouped according to their similarity. In normal cases, if there are two types of data, they will be clustered into two groups and using the hyperplane, the data will be dichotomously as a large as possible from the plane [40].

**Step 5:** With the algorithms mentioned above, there are a variety of n-gram methods for the algorithms, including unigram, bigram, trigram, mixes of unigram and bigram, or unigram and trigram as well as a mix of the three methods (unigram, bigram and trigram).

**Step 6:** The results are collected and compared with the results of the neural network and show the most accurate among them.

### 3.4 Machine Learning Classification Methods

In this part, we present detailed information about all the methods used in this work. The explanations cover all the rules and equations for the algorithms in the machine learning part of the work.

#### 3.4.1 Naïve-Bayes Algorithm

The Naïve-Bayes algorithm is one of the more widely used classifiers in the field of classification algorithms in general and in sentiment analysis specifically [26] [27]. It measures the probability of an event occurring with what is known as the Bayes algorithm, which plays a role in calculating the probability for the features of a label. A relation exists between the features of the label and this helps to measure the probability which can be extracted using the following equation:

$$P(\text{label}|\text{features}) = \frac{P(\text{label}) * P(\text{features}|\text{label})}{P(\text{features})}$$

Where *features* is related to *label*, and *P* is the probability of the feature to appear with this label. It is calculated as the probability of the occurrences of label multiplied by the

probability of the feature appearing associated with the label. It can be cast as the total number of occurrences of the label multiplied by the number of occurrences of the same label with a specific feature associated to it and divided by the total number of occurrences of the feature. To make the assumption for the features in the terms of naïve, we have the equation below:

$$P(\text{features}|\text{label}) \approx P(f_1|\text{label}) * P(f_2|\text{label}) * \dots * P(f_n|\text{label}) = \prod_{i=1}^n P(f_i|\text{label})$$

$$P(\text{label}|\text{features}) = \frac{P(\text{label}) * \prod_{i=1}^n P(f_i|\text{label})}{P(\text{features})}$$

Where  $f_i$  is the feature of the label as an individual feature. In more particular, the label might come with many features and each feature should be calculated separately from the other features of a label. With this assumption, the algorithm still produces an acceptable result and in some cases, a good result when the features and the labels are not very high. However, in the binary sentiment analysis, there are only two labels which make the probabilities limited to those labels. In the algorithm, the feature should be defined for the label and associate it with the suitable one as a result. This makes the interest of finding the probability unimportant and only the label should be the most important item on which we focus. In this case, there is a method called the MAP (maximum a posteriori), which estimates the label at a higher probability with the value  $label_{map}$ . The full equation is [29]:

$$label_{map} = arg \max_{label \in L} \left[ \frac{\hat{P}(\text{label}) * \prod_{i=1}^n \hat{P}(f_i|\text{label})}{\hat{P}(\text{features})} \right]$$

The probability of true features can be ignored and the new formula becomes:

$$label_{map} = arg \max_{label \in L} \left[ \hat{P}(\text{label}) * \prod_{i=1}^n \hat{P}(f_i|\text{label}) \right]$$

The  $P$  is converted to  $\hat{P}$  power because the result should be estimated to its label from the training [29].

The numerous multiplications of the probabilities are generated from the previous equation. This can easily lead to a decimal value and to overcome this problem, there is a need to use a logarithm, thus:

$$\log(XY) = \log X + \log Y$$

Here, the summation of the logarithm is the multiplication of the probabilities and since the function is a routine function, it is applied to the two parts of the equation. As a result, the gained value will be the maximum value. Since the only value that changes is a numeric value, it does not affect the final result of the label. The equation after adding the logarithm function is:

$$label_{map} = arg \max_{label \in L} \left[ \log \hat{P}(label) + \sum_{i=1}^n \hat{P}(f_i|label) \right]$$

Since  $P(label)$  is already defined for the training, the new  $P$  for the label will be:

$$\hat{P}(label) = \frac{N_{label}}{N}$$

Where  $N_{label}$  is the features associated with the label and  $N$  the number of the features from the training equation. To compute the conditional probability, we use:

$$\hat{P}(f_i|label) = \frac{F_{i \ label}}{\sum_{i' \in V} F_{i' \ label}}$$

Where  $F_{i \ label}$  is the number of appearances of the features in the training for a label and  $V$  is the total number of all the unique features for the same label to make the final Naïve-Bayes classifier. There is a need to cover the case when a new feature appears and it is not covered in the training. To prevent the conflict, it should disregard it and with this addition, the final formula for the label becomes:

$$label_{map} = arg \max_{label \in L} \left[ \log \frac{N_{label}}{N} + \sum_{i=1}^n \log \frac{F_{i \ label}}{\sum_{i' \in V} F_{i' \ label}} \right]$$

Based on [29], the Naive Multinomial Bayes algorithm is as presented in the figure below (Figure 7):

---

**Algorithm Naive Bayes:**

---

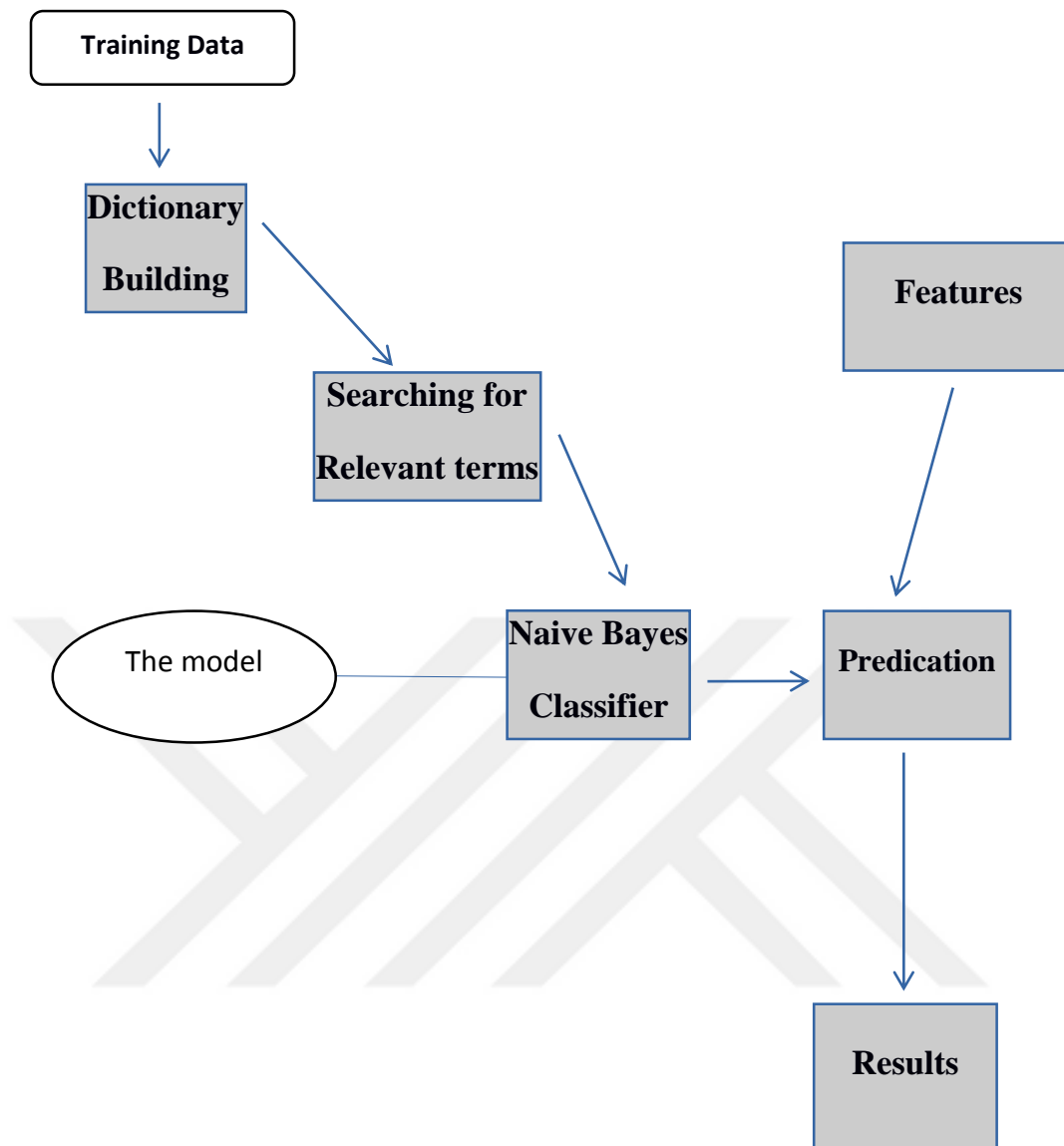
```
1 procedure TrainingNB(L, F)
2    $V \leftarrow \text{ExtractVocabulary}(F)$ 
3    $N \leftarrow \text{CountAllFeatures}(F)$ 
4   for each label  $\in L$  do
5      $N_{\text{label}} \leftarrow \text{CountFeaturesBelongingToLabel}(F)$ 
6      $\text{priorProb}[\text{label}] \leftarrow N_{\text{label}} / N$ 
7      $\text{text}_{\text{label}} \leftarrow \text{CombainTextOfAllFeaturesBelongingToTheLabel}(F, \text{label})$ 
8     for each  $f \in V$  do
9        $F_{i\text{label}} \leftarrow \text{CountFrequencyOfFeatures}(\text{tex}_{i\text{label}}, f)$ 
10      For each  $f \in V$  do
11         $\text{condProb}[f][\text{label}] \leftarrow \frac{F_{i\text{label}}}{\sum_{i \in V} F_{i\text{label}}}$ 
12      end for
13    end for
14  end for
15  return  $V, \text{priorProb}, \text{condProb}$ 
16 end Procedure
17 procedure TestingNB(L, V, priorProb, condProb)
18    $W \leftarrow \text{ExtractFeatures}(V)$ 
19   for each label  $\in L$  do
20      $\text{score}_{\text{label}} \leftarrow \text{logpriorProb}[\text{label}]$ 
21     for each  $f \in W$  do
22        $\text{score}_{\text{label}} += \text{logcondProb}[f][\text{label}]$ 
23     end for
24   end for
25   return  $\text{argmax}_{\text{label} \in L} \text{score}[\text{label}]$ 
26 end procedure
```

---

**Figure 7:** Naïve-Bayes algorithm

To conclude, the final explanation about the system works with the Naïve-Bayes classifier for sentiment analysis (see Figure 8). The dataset requires labeled training data that are fed into the system as the inputs. Then the features are extracted and injected into the classifier, followed by the system classifying the data. Finally, the data are tested to measure the accuracy of the system.





**Figure 8:** Estimation model for the Naive-Bayes classifier for sentiment analysis

### 3.4.2 Maximum Entropy Algorithm

The maximum entropy classifier uses data as experimental inputs. Many researchers have found that the Maximum Entropy algorithm produces better results than the Naïve-Bayes classifier in their works [38] [41] [42]. The main difference between it and the Naïve-Bayes classifier is that the assumptions are made independently for the word occurrences. The probability in the Maximum Entropy for a word is maximization of the unique distribution of the words. The algorithm takes words as inputs and compares them with a set of words which are features and are related to a document as well as by checking whether it belongs to the document and labeled with “1” for

availability and “o” for the opposite. The probability can be measured with the following equation:

$$P(c_j|d) = \frac{1}{Z(d)} \exp(\sum_i \lambda_i f_i(d, c_j))$$

Where  $P(c_j/d)$  is the probability of a class to appear in a document and  $Z(d)$  is the summed values of the iterated value of  $c_j$ . The equation works with what is known as GIS (Generalized Iterative Scaling) [43]. It converges the values with uniform distribution iteratively until it reaches the maximum entropy value.

### **3.4.3 Stochastic Gradient Descent**

The SGD is a linear model for document classification into either positive or negative. This model is called binary and when there are more than labels, such as natural labels, and then the model will contain three linear models, one for each label. This method decreases the loss in the model for the loss function. The method updates the weights at each new input with regard to the training and is therefore called a “stochastic.” This manner of updating makes the algorithm fast in terms of training and it helps in making the combinations of features [44].

---

**Algorithm 1** Stochastic gradient descent with hinge loss and elastic net regularization

---

```

1:  $t \leftarrow 1/(\eta \alpha)$ 
2:  $u \leftarrow 0$ 
3: Initialize  $w_j$  and  $q_j$  with 0 for all  $j$ 
4: for  $epoch$  to  $N_{ITER}$  do
5:   for  $i$  to  $N_{SAMPLES}$  do
6:      $s \leftarrow w^T x^{(i)}$ 
7:      $\eta \leftarrow 1/(\alpha t)$ 
8:      $c \leftarrow \text{CLASSWEIGHT}(y^{(i)})$ 
9:      $u \leftarrow u + ((1 - \rho) \eta \alpha)$ 
10:    for  $j$  to  $N_{FEATURES}$  do
11:       $\frac{\partial \ell}{\partial w_j} \leftarrow \begin{cases} -y^{(i)} x_j^{(i)} & \text{if } y^{(i)} s < 1 \\ 0 & \text{otherwise} \end{cases}$ 
12:       $w_j \leftarrow (1 - \rho \eta \alpha) w_j - \eta c \frac{\partial \ell}{\partial w_j}$ 
13:       $z \leftarrow w_j$ 
14:      if  $w_j > 0$  then
15:         $w_j \leftarrow \max(0, w_j - (u + q_j))$ 
16:      else if  $w_j < 0$  then
17:         $w_j \leftarrow \min(0, w_j + (u - q_j))$ 
18:         $q_j \leftarrow q_j + (w_j - z)$ 
19:     $t \leftarrow t + 1$ 

```

---

**Figure 9:** SGD Algorithm

### 3.4.4 Support Vector Machine Algorithm

The SVM is a classification algorithm for both linear and non-linear data. Basically, it takes a set of points as data for labels and associates features to those labels and classifies them depending on their similarity to the main labels. The main objective in the algorithm is to separate the data into two groups of data in terms of being positive or negative. The SVM equation is:

$$f(x) = \sum_{i=1}^n a_i k(x, x_i) + b$$

Where  $k$  is the exponential, which can be extracted with the following formula:

$$k(x, x_i) = \exp\left(-\frac{\|x - x_i\|^2}{2\sigma}\right)$$

The SVM algorithm can deal with isolating the data on the linear and non-linear and even overlapping data by producing many versions of the hyperplane. However, since

the data are binary, the process will be linear without any of the complexity of the non-linear or multi-dimensional [45].

### **3.5 Deep Learning Method**

In this part, all the details about the neural network method used in this work are presented and here our model will be built with the RNN (Recurrent Neural Network). The Word to Vector method is also explained in this part.

#### **3.5.1 Word to Vector (word2vec)**

This machine learning method deals with the simple techniques for data and weighting them with regard to occurrences of the terms in the documents or with similar techniques. This can make the work very limited in terms of the vocabulary size and especially when there are high-dimensional data. In 2013, Mikolov [31] developed a model to convert words into numerical representations, which makes dealing with them easier and increases the accuracy of the tests for Natural Language Processing applications. Words are represented as a vector of numbers, which is the reason for the term *word to vector*. Moreover, it is also known as word embedding. Those vectors, or embeddings, are used to map words with their associated numeric representations. The length of the vectors is a fixed size for all the terms, called a “dimension,” and built corresponding to the calculation of the building from the gathered data used to generate those embedding's. The booming of the word to vector is a return to the fact that the embeddings can capture the semantic meanings of the words. For example, the word *Rome* is close in similarity to the vector of the word *Italy* and the similarity between them will be like 0.7 as a score, while the *Rome* vector and the *Car* vector will receive a similarity of 0.2 as a score. This makes the predication better and reduces the gap between it and the human mind for the measurement of the similarity between words. Also important with regard to embedding's is that this method is an unsupervised technique and it can be built with a corpus of texts rich with words, including Wikipedia. Mikolov's Word2Vec is the most well-known embedding algorithm and it has two different architectures (Figure 10, namely CBOW (Continuous Bag of Words) and Skip-Gram. CBOW is good for predicting the next word that might appear after another word while Skip-Gram is good for predicting the words near another word regarding whether it came after or before it [31].

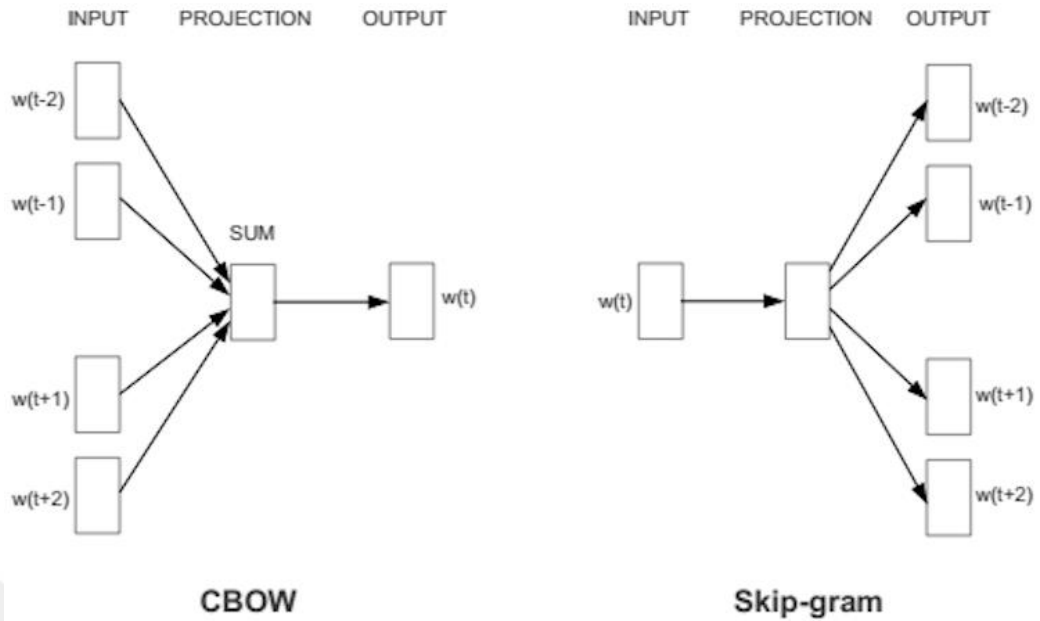
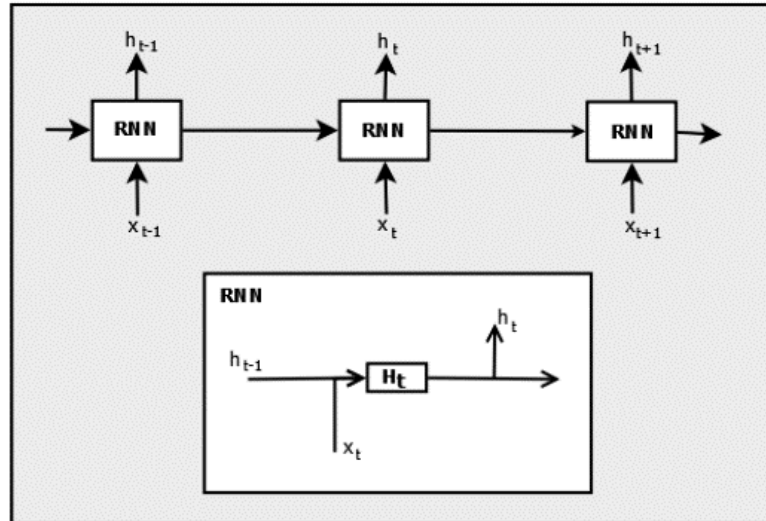


Figure 10: Mikolov's word2vec architecture

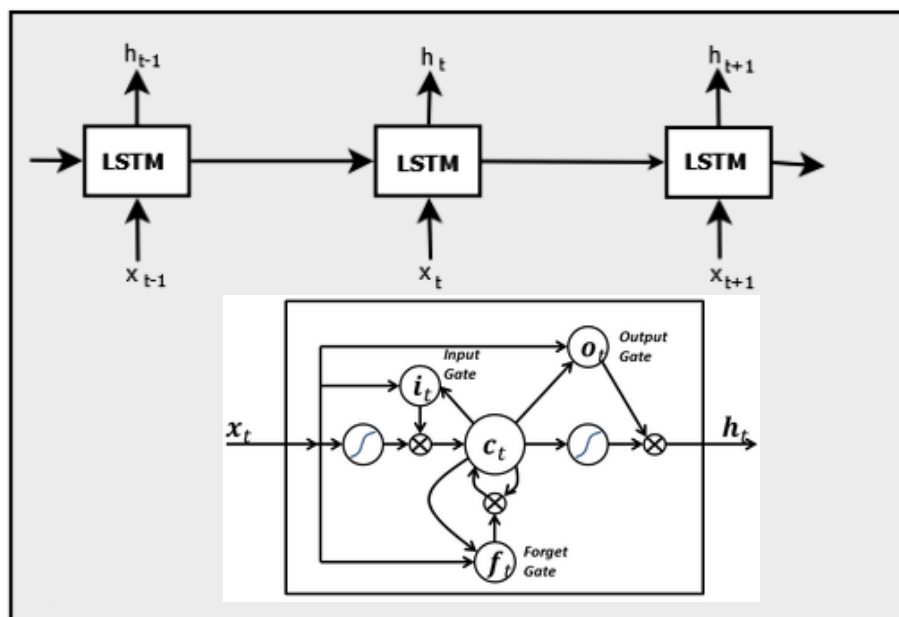
### 3.5.2 Recurrent Neural Network

RNN is a type of the neural network and it is called name returns for the reason that it deals with data sequentially, which makes it good when the data are texts. RNN can use data from previous entries and store them for calculations, which is another reason to make it good for dealing with texts. The problem of storing that information makes it limited to the few previous steps. The model acquires two entries at a time, one for the current entry and the other for the previous entry (see Figure 11). This makes it somewhat similar to the human brain such that it compares the text using the previous words having appeared previously.



**Figure 11:** Recurrent Neural Network

Due to the back propagation, the error rate might increase with this model when the model trains and updates the weight to minimize the error between the prediction and the correct value. To prevent this problem from appearing, the RNN has been developed with new features called LSTM (Long Short Term Memory). The key to the success of this method is the locking gates mechanism. This model stores the data from previous steps and has the ability to ignore data when they are not mandatory (see Figure 12) [46].



**Figure 12:** RNN with the LSTM model

### **3.5.3 Layers in Deep Learning Models**

In this section, all the layers used in the model will be explained in detail with their functionality and how they fit into the model.

#### **3.5.3.1 Embedding Layers**

As mentioned previously, the embedding is used to present the input in numerical form to make calculations easier and more accurate. There are many options to build this layer. It can be generated from a large corpus or by using pre-trained embeddings and making them static, which means it will not be updated during the training. The last option is the use of a pre-trained embedding and updating it during the training.

#### **3.5.2.2 Dropout Layers**

The main advantage of using this type of layer is to overcome the over fitting problem in the model. The model consists of many layers and those layers are connected and have the ability to deal with any number of high complexity inputs or outputs. This makes the model expensive in terms of calculations and the training data might affect the quality of the output due to its limitations. When this appears, it causes the model not to work well in the training and the quality of prediction will drop. The dropout layer prevents this scenario by making the model learn various copies of the same data to decrease the dependence of the data in the training.

#### **3.5.2.3 LSTM Layers**

LSTM is a better version of the RNN and it is widely used when dealing with texts. A benefit of using it is the enhancement of the accuracy of classification for the model. There are many underlying aspects of this performance. The first aspect is that the model can control the time of the data to enter the neuron. The next aspect is the ability to store any data from previous entries and using them to predict the data during subsequent steps. Finally, there is the ability to control what should or should not be output.

### **3.5.2.4 Activation Layers**

This is another layer in the model that applies a function which can be any type of activation function to generate an output. It helps the model to learn more complex calculations better than the linear regression and it is performed linearly. Any model that does not have an activation function will act with the function  $f(x) = x$  and it makes the model act like a single neuron and consume every network resource. For this reason, activation functions become important. In a particular manner, the activation function plays the main role in making the model extract a non-linear value with the help of the weights.

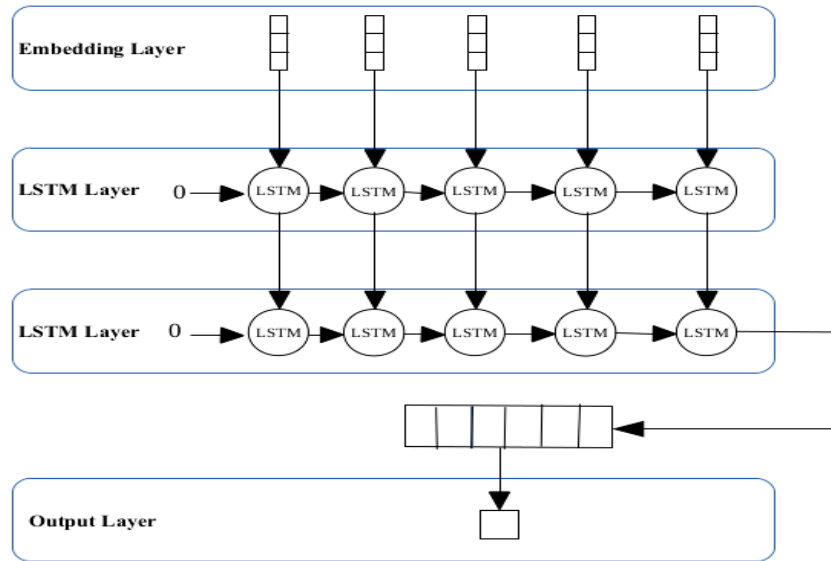
## **3.6 Network Architecture**

There are two types of LSTM networks: stacked and bidirectional. Here, we describe the details of those architectures.

### **3.6.1 Stack of LSTMs**

This model is a composition of two LSTM units with an activation layer and one embedding layer. The link between those layers is a recurrent connection and this makes it a deep network. This design is good at capturing the meanings in sentences in the higher LSTM layer, which in turn makes it very good at capturing the semantic meaning for the sentiment analysis task. (See Figure 13 for the stacked LSTM architecture.)



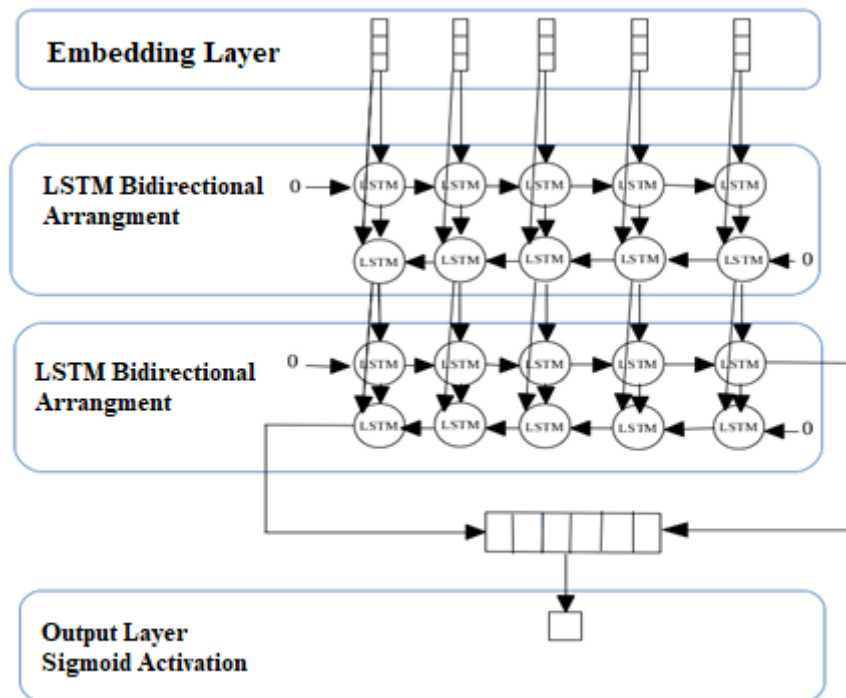


**Figure 13:** Stack of two-layer LSTMs

This model does not have a dropout layer. There is, instead, a dropout assumption and it works to update the training with a value of 0.25. For the embedding layer, there are many specifications, including the input dimension, input length, input weight and output dimension. The input dimension pertains to the number of words that will fit in the model at each entry as a maximum. The input length and weights are the maximum lengths for a sentence, such as the number of words that should be processed, and the weights are arrays to contain the weights for each word. Finally, the output dimension pertains to the embedding size. The activation function here is the sigmoid function and the weights are initialized with a zero value and are updated during the training.

### 3.6.2 Bidirectional LSTM

This architecture has been developed to increase the performance of RNN networks. Since a regular RNN does not have a forward step of the present state, the Bidirectional LSTM design fixes this by connecting all the current nodes to any output in the next layer, which helps to cover any past or future information to enhance the performance of the model. (See Figure 14 for Bidirectional LSTM architecture.)



**Figure 14:** Bidirectional LSTM architecture

It is clear in Figure 14 that the model has all of the stacked LSTM architecture. It has an embedding layer and a sigmoid activation layer. In addition, it has a dropout layer which makes it good to pass the overfitting problem. The main point here is that this provides the ability to move back and forth easily to update the information and capture the semantic meaning in the sentences.

### 3.7 Conclusion

We have covered all the details about the methods used in this work. The explanations covered the machine learning methods, including the Naïve-Bayes, ME, SGD and SVM algorithms. It also shows all the information about the deep learning method used in our thesis. The next chapter will show the design and implementation parts of this thesis.

## **CHAPTER FOUR**

### **Design and Implementation**

#### **4.1 Introduction**

In this chapter, we present all the information about the inner design and the mechanism of the models and algorithms used in our work and the implementation of the training and test parts for [35] and our model. Furthermore, we present information about all the tools used in the thesis.

#### **4.2 Tensorflow and Keras**

Tensorflow [47] is a machine learning library developed by Google. It is very powerful for high performance applications that require many numerical and complex computations. It can be used in many types of hardware, such as CPUs and GPUs and also in mobile devices. It is the main library in these experiments. Keras [48] is a library for machine learning and deep learning which is considered to be a high-level library that can be used on top of any other machine library such as Tensorflow, Theano, Caffe and many others. Keras has a large community of developers who return to the simplicity of the Keras coding, which might be a problem for many other libraries. For more exhaustive control over Tensorflow, we used Keras on top of it. The Tensorflow version used in this work was the 1.8 release.

#### **4.3 Machine Learning Methods**

The algorithms mentioned in the previous chapter were implemented with an n-gram model, which is also explained in the same chapter. In brief, the n-gram model is a method used for many applications, and from those we have the classification as one of those applications. It helps to predict the next term in a sequence, but in the field of

sentiment analysis, it helps to improve results when used especially with more than a unigram model. The sizes of the n-gram models used in this work are Unigram, Bigram and Trigram. Any model higher than 3-gram is called a high gram and designated with the number of grams followed by “gram,” such as four-gram, five-gram, six-gram and so on. Simply, the n-gram is a technique used to split sentences into many segments depending on the weight of the gram in the model. The gram here is used for the word as shown in the example sentence “*The day was shinny and beautiful*”. When the model is a unigram model, the sentence is divided as follows: “*The*”, “*day*”, “*was*”, “*shinny*”, “*and*”, “*beautiful*”.

For the Bigram and Trigram:

1. Bigram model: When two words are used for the model. “*The day*”, “*day was*”, “*was shinny*”, “*shinny and*”, and “*and beautiful*”.
2. Trigram model: When the gram has three words. “*The day was*”, “*day was shinny*”, “*was shinny and*”, and “*shinny and beautiful*”.

#### **4.4 Deep Learning Model**

The neural network used in this study is the RNN (Recurrent Neural Network) with LSTM (Long Short Term Memory) as mentioned earlier in the previous chapters. We covered all the details about the model and the supplements used with it. Moreover, we covered all the details about the mechanisms of the model and finally presented all the results obtained from our work.

##### **4.4.1 Word2vec Model**

We explained the word2vec model implemented by Mikolov and covered the details about it in the third chapter. In our work, we used another word vectorizing model called GLOVE, which stands for *Global Vectors for Word Representation*. It was implemented at Stanford University by Jeffery Pennington, Richard Socher and Christopher D. Manning. GLOVE is another algorithm for word numeric representation and its unsupervised learning method. It uses the co-occurrences of words in a corpus and gathers the statistics of the word and represents them in a numerical form for use in tasks. In [49], GLOVE showed better results in comparison with Mikolov’s [31]

word2vec in the analogy task. Performance decreased with the negative samples for both the CBOW and Skip-gram models and GLOVE outperformed them and showed better results with faster training times, as shown in Figures 15 (a) and (b).

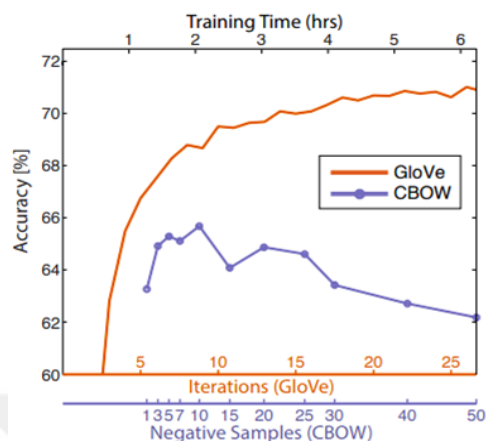


Figure 15 (a) Glove vs. CBOW [49]

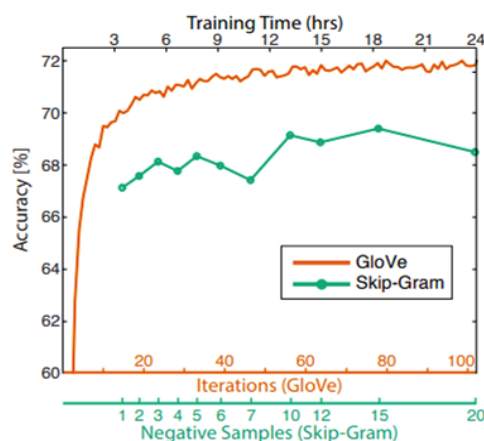


Figure 15 (b) Glove vs. Skip-gram [49]

We downloaded the pre-trained word vectors from [49]. Because training word vectors requires a large corpus such as *Wikipedia* and high performance hardware, we might have the same results as the pre-trained vectors at the end. The vectors we used were collected using the GLOVE algorithm from the corpus data of both *Wikipedia* 2014 [50] and *Gigaword* (5th edition) data [51]. The data were collected from 6 billion tokens and contained a vocabulary of 400,000 unique words with vectors of different dimensions (50d, 100d, 200d and 300d). We used other data collected from 42 billion tokens and 1.9 million unique vocabulary items with 300 dimensions. The word vectors were used as weights for the words in our model.

#### 4.4.2 The RNN and LSTM Mechanisms

The main feature in the RNN is the sequential design which makes it one of the best architectures in terms of text processing. There are many types of RNN and the most recognizable is the LSTM, which is good for storing information from previous steps and using that information in following steps to generate more accurate data for classification or prediction. Since the RNN is sequential, it has the input feed into the network consecutively, known as time steps (see Figure 16).

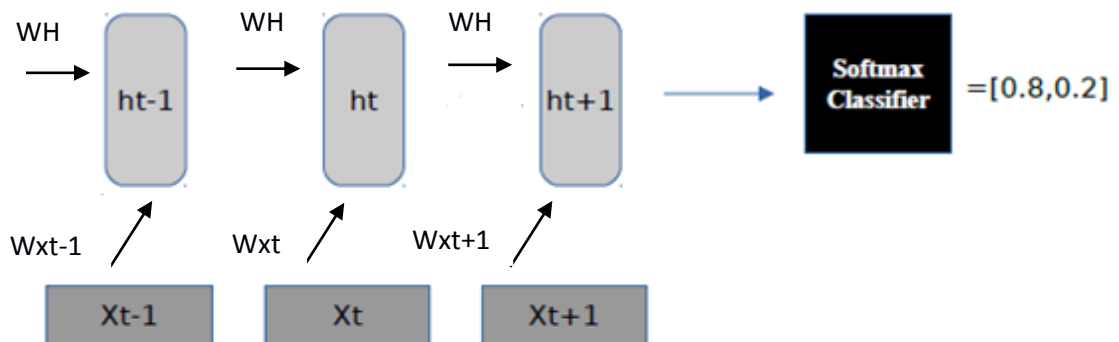
The	movie	was	...	outstanding
X0	x1	x2		x19
t=0	t=1	t=2		t=19

**Figure 16:** Time step inputs feeding in RNN

The model requires a weight for the inputs which are the vectors that are generated from the GLOVE algorithm. The weight is a numerical matrix representing the word and this design is used to express the analogy of the words associated with the semantic meaning of the words. The hidden layer in the model uses a sigmoid function to process the input and the output of the previous steps and totals them in one value for the output of the current state (see Equation (1)).

$$h_t = \sigma(W^H h_{t-1} + W^X x_t) \quad (1)$$

where  $h_t$  is the output of the current state,  $W^H h_{t-1}$  is the output vector generated from the previous steps and  $W^X x_t$  is the vector of the current state.  $W^H$  is a weight matrix that remains static throughout every step and  $W^X$  is the weight of the current input which changes every time depending on the input. The formula helps the next hidden state with taking the decision for the current processing. The output may remain the same as the previous steps if the neuron determines that it is not important for the overall summary of the text. In other words, if the value of  $W^H$  is greater than the  $W^X$ , then the input  $x$  in the current state is unaffected and it passes the previous vector as an output to the next step. The weight matrix is updated through time with the update of the optimizer, or what is known as back propagation. The final step in the hidden state is the sigmoid function, which yields a value between 0 and 1 to decide the sentiment classification (see Figure 17).



**Figure 17:** Sequential process in RNN

To gain a clear understanding of the LSTM and how to store the information and ignore the data with an example, we have the following text:

Text: “the big brother is 30 years old, the day was rainy, the younger brother is 27 years old”

Question: “What is the difference in years between the two brothers?”

We can see clearly that the sentence consists of three parts; two are associated with the age issue, while the middle part is far in terms of subject from the question. The main part is to separate the data and feed them into the LSTM to handle the process and extract the data associated with the question. Firstly, we can look at the architecture of the LSTM (Figure 18).

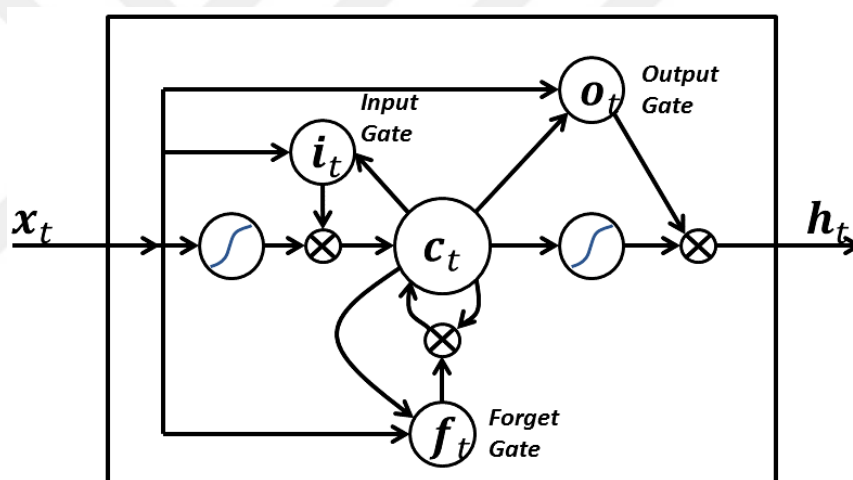
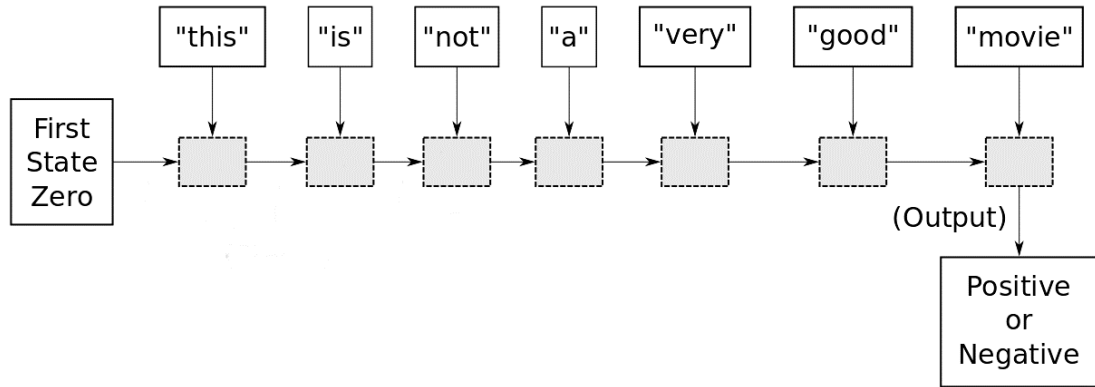


Figure 18: LSTM unit architecture [46]

The input is entered as  $x_t$  and depending on the previous  $h_{t-1}$ , it will be decided by the  $c_t$  as to whether this information is related to the question. Basically, the unit will initially ignore most of the data until it stores the word “30,” then it will continue to parse all the data and ignore them with the forget gate until it stores the next input which is similar to the previous input (a numeric input), namely the word “27.” Finally, it will take only those data that are related to the question and ignore the other data.

Another example for classification is when a sentence is entered into the LSTM layer for sentiment analysis. It focuses on the key words that have sentiment as being either positive or negative (see Figure 19).



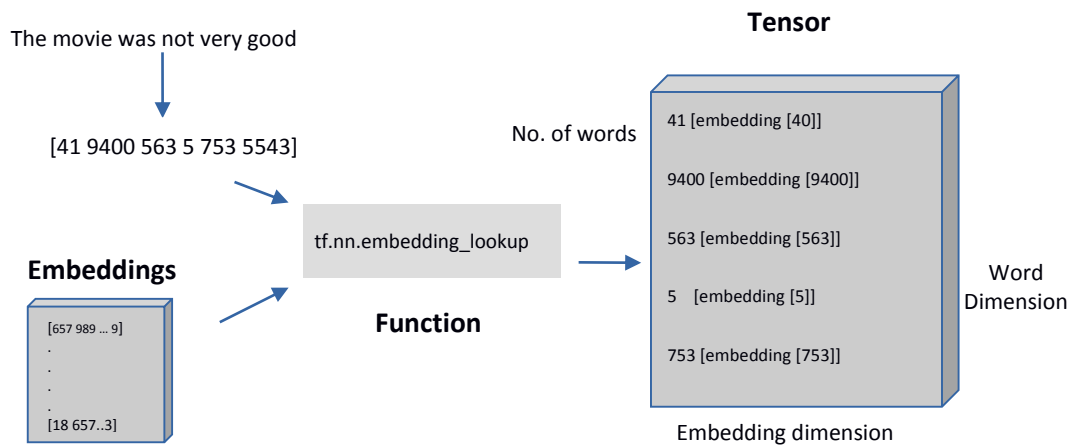
**Figure 19:** Sentiment analysis for a sentence

In the figure above, the model ignores every word until it reaches the word “not” and stores the vector for the word and passes it to the next unit. The layer will ignore the words until reaching the word “good.” Finally, the sigmoid activation function will summarize the overall score between 0 and 1 to give the final sentiment for the word as being either positive or negative.

#### 4.4.3 Neural Network Flowchart

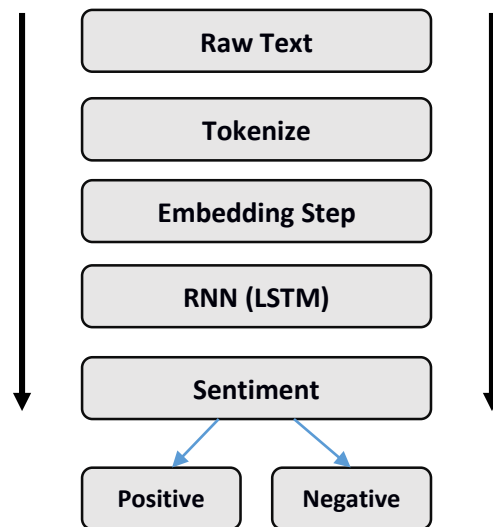
In our experiments, we feed the reviews as raw text into the model. Then we use a tokening method to convert the words into integers depending on the order of the words in all 25,000 reviews. Next, we convert the data into vectors using the vectors from the GLOVE pre-trained embeddings. We only used the 1,000,000 most frequent words for the analysis. Tensorflow builds the first tensor with a dimension of 3 values: the number of the words in the review, the dimension of the embedding and the dimension of the word which is 1 using the lookup function in Tensorflow (see Figure 20).





**Figure 20:** Building the tensor step in TensorFlow

In order to decrease the cost of processing, we converted every review and stored them as a numpy array of reviews with a length of words, which were stored under the name “*idsMatrix.npy*”. Figure 21 shows all the steps of the model that we followed in the process.



**Figure 21:** Sentiment analysis flowchart

#### **4.4.4 How It Works**

The steps of the process in the model are in two parts: training and testing. The training steps occur when the network learns how to measure the sentiment by training itself with labeled data to learn the pattern of the sentiment. The model receives the training data as reviews and those reviews are labeled as either negative or positive. The network takes a review and processes it to learn the pattern, after which it stores that pattern with its own label. After each step, it updates the weights of the network by learning the best parameters for the layers. To avoid the over-fitting problem, we separated the training data into two parts: training and validation. We segmented the reviews as 90% training and 10% validation. For example, if we feed 100 reviews each time into the batch, then 90 reviews will be used for training and learning the pattern in that step, and 10 reviews will be used to test the pattern learned in this step. If the values of the validation test continue to appear with the same accuracy, it then means that we have an over-fitting in the model. In this case, we need to stop the training and change the parameters of the network, such as the number of LSTMs in the hidden units and batch size, etc. If the training continues and the validation accuracy value continues to change, then we wait until the model trains and updates itself with the learning rate to handle the reviews with a more accurate analysis. What is actually happening underneath the model is that the network learns the vector of the review and stores it with the associated label and whenever it receives a review similar to its vector, it then assigns it the same label. After the model finishes the training, the model should be tested with new reviews that were never used in the training or validation to test the accuracy of the model. The last accuracy determines the quality of the sentiment analysis task of the network. If the accuracy sufficiently good, then the model needs modification within the layers and design. The test step is concluded by feeding the model with the test review and analyzing it. Then the sentiment of it is predicted followed by comparing the result with the label associated to this review. If it is correct, it counts as a correct prediction, otherwise it is a false prediction. The final accuracy is the percentage of the total number of correct predictions of all the test reviews.

#### **4.4.5 Neural Network Parameters**

Before we provide the final results of the work, there is a need to understand the parameters used in this work and the details for each one of those parameters. The first

detail of the parameters is the batch size, which is a term always used with neural networks indicating the number of samples that will be passed to the model to process at one time. In particular, if we have 100 samples that need to be tested in the model, then it will be entered as 10 samples each time until the network completes every sample. The benefits of the batch size include decreasing the memory required, which means less information to store in the memory. Moreover, with the smaller size of the batch, the network will train faster.

The next parameter is the max sequence length, which simply means the maximum number of words of the review to be injected for the training. The LSTM units are the next parameters which are assigned for the number of LSTMs in the hidden state. If we have 20 words and 5 LSTMs, then each group of 5 words will be covered by the five LSTMs and only the important information will be stored before it receives the next 5 words.

Epoch is another parameter in our model. 'Epoch' is a term used widely in neural networks meaning the number of times the samples will be added to the network for processing. Each time the network processes the data and completes doing so, it is then considered to be an epoch. With many epochs, the network will process the data many times to decrease the error rate in the training.

The last of the parameters is max words, which relates to the words used in the process. After the tokening step, the words will be sorted according to their frequencies and the first word becomes the most important word among the other words and the next one will be less important with fewer appearances. The same applies to the remaining words.

Since there are many word vectors used for the training with a high number of words in each embedding, there was a need to decrease the time that can be wasted on the words occurring at lower frequencies. Max words limits the number of words needed in the tasks. These are the important parameters that require being known for the modifications occurring in our model.

## **4.5 Conclusion**

In this chapter, we have explained the designs and mechanisms of the methods used in the practical part. We also covered the details about the parameters of the network. Moreover, we added an explanation about the training and testing steps during the work. In the next chapter, we present the conclusion we reached for the final result of each method from the previous work and we compare them with our method results.



## **CHAPTER FIVE**

### **Results and Discussion**

#### **5.1 Introduction**

In this part of the thesis, all the results obtained from the algorithms used in the thesis and analysis of the results of the machine learning approach algorithms and the neural network results are presented. The work was performed with the IMDB dataset. The results of the machine learning methods have already been carried out by [35]. The neural network experiments occurred with Tensorflow and the Python programming language. The accuracies of each algorithm are presented and compared with our method. As a conclusion, the results will be discussed according to the information and the gathered data from the experiments.

#### **5.2 Results of Machine Learning Methods**

The results of the methods used in [35] are covered in the order of the algorithms: NB, ME, SVM and SGD. The authors also have mentioned in the literature section all the related works using the same methods and datasets to compare the enhancements they made. Moreover, they compared the achieved results of previous works with the results they gained in their work (see Table 3).

**Table 3: Sentiment Analysis Works Comparisons**

Paper	Method used	Algorithms	Dataset	Results
Pang 2002	Classify the data with different methods using different n-gram models	NB, ME, SVM	IMDB dataset	SVM with Unigram: 82.9, ME with Bigram: 77.4 SVM with Unigram+Trigram : 82.7
Salvetti 2004	Using machine learning methods with OvOP (overall opinion polarity)	NB, Makarov Model (MM)	IMDB dataset	NB : 79.5, MM : 80.5
Beineke 2004	Features from linear combination	NB	IMDB	NB : 65.9
Mullen & Collier 2004	Classification model combined with words with assigned values	SVM	IMDB dataset	SVM : 86.0
Dave 2003	Several machine learning algorithms	SVM lite, ML using Rainbow, NB	Cnet dataset, Amazon dataset	NB : 87.0
Matsumoto 2005	Document level classification with the syntactic relationships among the words	SVM	IMDB dataset	Unigram : 83.7, Bigram : 80.4, Unigram+Bigram: 84.6
Zhang 2015	SVM perf using word2vec model	SVM perf	Lexicon based model: 89.95, POS model: 90.30	Comments of Chinese products (clothes)
Liu & Chen 2015	Using 11 multi-label with different aspects	Eight evaluation matrices	DUTSD (Dalian University of technology sentiment dictionary), NTUSD (National taiwan university sentiment dictionary)	AVG highest precision : 75.5
Luo 2016	Machine learning techniques from (Ekman & Friesen 1971)	SVM, NB, Decision Tree (DT)	Dataset from the lion forum	SVM: 78.31, NB: 63.28, DT: 79.21
Niu 2016	Classification with statistical learning methods	Bag of words with TF, TF-IDF	Twitter data	Text: 71.9, visual feature: 68.7, multiview: 75.2

As can be seen from the table above, many works depend on the n-gram model for sentiment classification. Most of the similar papers achieved a good result with the unigram model in many cases; however in other cases, it failed the accuracy and became poor. As an example of this type of failure, for the sentence “the item is not good” with the unigram model, it gave the result of the review as being natural and returned a positive polarity for the word “good” and a negative polarity for the word “not,” so the correct classification is a negative review. However, with the bigram model, the classification becomes negative due to the negative classification of “not good,” thereby making it a correct classification. With this study and comparison, the authors

suggested that with higher numbers than unigram in the n-gram model, the accuracy might increase. Another problem appears with the papers that used the POS (part of speech) tags for classification. Here, the authors observed that the POS tag was not accurate and it changed depending on the context. For example, the word “Book” is a noun tag, but in case the word comes as a verb tag, such as “ticket booking,” this would result in a confusing classification. To overcome this problem, each word was considered separately without the POS tags method. Many papers agree that texts must be converted into numerical form to deal with them and the best representation in this case is a numerical matrix. Most of them used the TF or the TF-IDF method to convert texts into numbers. The researchers created a combination of TF-IDF and the Count Vectorizing methods.

**Table 4:** Results of Methods Used in [35]

Algorithm		Results
Naive Bayes Classifier	Unigram	83.652
	Bigram	84.064
	Trigram	70.532
	Unigram, Bigram	86.004
	Bigram, Trigram	83.828
	Unigram, Bigram, Trigram	86.232
Maximum Entropy	Unigram	88.48
	Bigram	83.228

	<b>Trigram</b>	71.38
	<b>Unigram, Bigram</b>	88.42
	<b>Bigram, Trigram</b>	82.948
	<b>Unigram, Bigram, Trigram</b>	83.36
<b>Support Vector Machine</b>	<b>Unigram</b>	86.976
	<b>Bigram</b>	83.872
	<b>Trigram</b>	70.204
	<b>Unigram, Bigram</b>	88.884
	<b>Bigram, Trigram</b>	83.636
	<b>Unigram, Bigram, Trigram</b>	88.944
<b>Stochastic Gradient Descent</b>	<b>Unigram</b>	85.11
	<b>Bigram</b>	62.36
	<b>Trigram</b>	58.408
	<b>Unigram, Bigram</b>	83.36
	<b>Bigram, Trigram</b>	58.744
	<b>Unigram, Bigram, Trigram</b>	83.33

Table 4 presents the result of the experiments. We can easily observe that the best result of the single level n-gram model is the result of the bigram model which returns the probabilistic model for the Naïve-Bayes algorithm and the dependency of the features. Hence, the result of the trigram model is poor compared with the unigram and bigram



models. This paradox occurs with the trigram model when a word is repeated many times, such as “*day was shinny*” and the repetition of the word “*shinny*” in “*shinny and beautiful*.” If the same case appears in a sentence with the word “not,” a different classification can be made. Moreover, even combinations in the trigram cause poor performance when used with the bigram model. The best results achieved with the Naïve-Bayes algorithm was when the n-gram was unigram + bigram + trigram.

With the Maximum Entropy method, the result of the unigram model was the best among all the n-gram models used in the experiments. This result was achieved due to the dependence of the algorithm on the conditional distribution of the words. The performance decreased with the other models due to the number of appearances of the word with different polarities depending on the new distribution from the models higher than a unigram model. Moreover, this continued to cause problems even with combinations of the n-gram models.

As presented in the SVM results, since the support vector machine is a non-probabilistic linear classifier, the best result among the one level n-gram model is the unigram model. The SVM algorithm depends on training the model to discover the best hyperplane to classify the data, which causes problems when two or more words appear as one point and cause confusion. The combination of the model achieved better results when the unigram model was used in it.

The stochastic gradient descent is an estimation of the gradient for a review depending on the learning rate to mitigate any loss in the classification. The authors found that the result is better in the unigram model than in the bigram and trigram models. In particular, it was high when it selected one random word for the analysis and decreased with the increase of the words randomly selected. Even with the combinations, the result of the unigram joined with the bigram was the best among the other combinations.

As a final conclusion of the performance of the accuracy for the algorithms compared with the previous works, the authors showed that the method used in the previous works provided a comparison and conclusion for the method used in their work and not appearing in previous works. Table 5 shows the results attained by the previous researchers and the results of [35]. Those papers used the n-gram model.

**Table 5:** Comparison of Methods Used in IMDB Sentiment Analysis

Algorithm		Pang	Salvetti	Beineke	Mullen & Collier	Matsumoto	Paper Results
Naive-Bayes Classifier	Unigram	81.0	79.5	65.9	-	-	83.65
	Bigram	77.3	-	-	-	-	84.06
	Trigram	-	-	-	-	-	70.53
	Unigram, Bigram	80.6	-	-	-	-	86
	Bigram, Trigram	-	-	-	-	-	83.82
	Unigram, Bigram, Trigram	-	-	-	-	-	86.23
Maximum Entropy	Unigram	80.4	-	-	-	-	88.48
	Bigram	77.4	-	-	-	-	83.22
	Trigram	-	-	-	-	-	71.38
	Unigram, Bigram	80.8	-	-	-	-	88.42
	Bigram, Trigram	-	-	-	-	-	82.94
	Unigram, Bigram, Trigram	-	-	-	-	-	83.36
Support Vector Machine	Unigram	72.9	-	-	86.0	83.7	86.97
	Bigram	77.1	-	-	-	80.4	83.87
	Trigram	-	-	-	-	-	70.16
	Unigram, Bigram	82.7	-	-	-	84.6	88.88

	<b>Bigram, Trigram</b>	-	-	-	-	-	83.63
	<b>Unigram, Bigram, Trigram</b>	-	-	-	-	-	88.94
<b>Stochastic Gradient Descent</b>	<b>Unigram</b>	-	-	-	-	-	85.11
	<b>Bigram</b>	-	-	-	-	-	62.36
	<b>Trigram</b>	-	-	-	-	-	58.40
	<b>Unigram, Bigram</b>	-	-	-	-	-	83.36
	<b>Bigram, Trigram</b>	-	-	-	-	-	58.74
	<b>Unigram, Bigram, Trigram</b>	-	-	-	-	-	83.36

We can easily observe that Pang used the machine learning methods of the Naïve-Bayes algorithm, Maximum Entropy and the support vector machine with the n-gram models which covered the unigram, bigram and a combination of them. Salvetti and Beineke implemented the models for sentiment analysis with the Naïve-Bayes classifier and n-gram model. However, they only covered the unigram model. Mullen and Collier used the Support Vector Machine method with a unigram model. Matsumoto extended the work by adding the bigram and a combination of the unigram and bigram for Sentiment Analysis on the IMDB dataset. The researchers in [35] used the same method with an additional algorithm, namely the stochastic gradient descent algorithm, and the n-gram models. Moreover, the n-gram models used in the study were extended by covering the trigram model and the combined models were also increased by adding the combination of bigram and trigram and unigram, bigram and trigram. The results achieved in the work were better than the result of the literature and the methods covered many n-grams models. In conclusion, the authors stated that the use of the n-gram models with increased n values would produce better results. Only when the value exceeded two grams, the accuracy dropped. Although some papers suggested the use of the POS tag might increase the accuracy of classification, the authors claim that it might cause confusion in the work, which makes the model inaccurate. Instead, they used a combination of different n-gram models, which produced better results.

### 5.3 Deep Learning Results

In our neural network, we used many parameters and made many modifications in the model in addition to much tuning until we achieved our best result. Since most neural networks, especially the deep learning networks, perform a high number of computations, there needs to be a great deal of tuning as well as high-performance hardware to supply the cost of computation needed in the model to achieve the best results and final form of the network.

Firstly, we started the work with the training and finally with the testing. We used in our work the following parameters:

- Batch size: 100
- Embedding: GLOVE of 6 billion words of dimension 50
- Max number of words: 1,000,000 words
- Max length of review: 2,500
- Number of LSTM units: 64
- Epochs: 10

With those parameters we started our first training and testing and the result was 84.06% for the testing with the 25,000 reviews being dedicated to the test issue. We made the first tuning for the characteristics of the model with the following new parameters:

- Batch size: 100
- Embedding: GLOVE of 6 billion words of dimension 50
- Max number of words: 1,000,000 words
- Max length of review: 2,500
- Number of LSTM units: 60
- Epochs: 10

We assumed that the number of LSTMs would be high and the memory dependency of the LSTM had a large gap. Then we decreased it to 60. With this number of LSTMs, the enhancement was hardly better with an accuracy of 84.311%. Then we made an attempt with another modification, but this time using a higher number of dimensions for the embedding than previous number of LSTMs with these parameters:

- Batch size: 100
- Embedding: GLOVE of 6 billion words of dimension 100
- Max number of words: 1,000,000 words
- Max length of review: 2,500
- Number of LSTM units: 64
- Epochs: 10

On the third attempt, we achieved a better result with 86.78% accuracy. With this new result, we commenced another task at a higher number of dimensions for the word vectors at 200 dimensions of embedding and the parameters became:

- Batch size: 100
- Embedding: GLOVE of 6 billion words of dimension 200
- Max number of words: 1,000,000 words
- Max length of review: 2,500
- Number of LSTM units: 64
- Epochs: 10

This time the accuracy also increased to 88.83%. Each time we used an embedding of a higher dimension, the accuracy increased. On this attempt, we applied the highest dimension of embedding with the parameters below:

- Batch size: 100
- Embedding: GLOVE of 6 billion words of dimension 300
- Max number of words: 1,000,000 words
- Max length of review: 2,500
- Number of LSTM units: 64
- Epochs: 10

This time the result was very good and the highest among all the previous tests with an accuracy of 89.36%. With this accuracy, we returned to the first modification we made for the number of LSTMs and performed another test with the following parameters:

- Batch size: 100
- Embedding: GLOVE of 6 billion words of dimension 300
- Max number of words: 1,000,000 words
- Max length of review: 2,500
- Number of LSTM units: 60
- Epochs: 10

The result was 89.72% accuracy. On this attempt, we abandoned decreasing the numbers of LSTMs for the current time and we started with another embedding. However, this time we performed another test with Mikolov's word2vec that was

collected using the Google News website before it closed; it is currently available on the Internet. The new parameters were:

- Batch size: 100
- Embedding: word2vec of Google News 100 billion words of dimension 300
- Max number of words: 1,000,000 words
- Max length of review: 2,500
- Number of LSTM units: 64
- Epochs: 10

The accuracy obtained this time was 86.30%, which was lower than in the previous test. Before we left the Mikolov embedding, we gave it another opportunity with a smaller number of LSTMs with the following parameters:

- Batch size: 100
- Embedding: word2vec of Google News 100 billion words of dimension 300
- Max number of words: 1,000,000 words
- Max length of review: 2,500
- Number of LSTM units: 60
- Epochs: 10

This time, the result was enhanced a little, with an accuracy of 87.41%, after which we left this embedding and returned to GLOVE. However, this time with a higher number

of LSTMs, we started to increase it with a small number of increments until we reached a new result with the following parameters:

- Batch size: 100
- Embedding: GLOVE of 6 billion words of dimension 300
- Max number of words: 1,000,000 words
- Max length of review: 2,500
- Number of LSTM units: 100
- Epochs: 10

The result this time became 90.03% for the first time. This remained with one final modification. We started with another GLOVE embedding which was generated from a larger corpus. This time the GLOVE we used was collected from a corpus with 42 billion words and a smaller number of LSTMs with the following parameters:

- Batch size: 100
- Embedding: GLOVE of 42 billion words of dimension 300
- Max number of words: 1,000,000 words
- Max length of review: 2,500
- Number of LSTM units: 64
- Epochs: 10

The accuracy increased by a very small amount to become 90.34% accuracy. The only item remaining for us this time was to increase the LSTMs again with small parts. The best results we achieved were at 100 LSTMs and the number of epochs. The best result was achieved with 13 epochs. The result was 91.04% accuracy with the following parameters:

- Batch size: 100
- Embedding: GLOVE of 42 billion words of dimension 300



- Max number of words: 1,000,000 words
- Max length of review: 2,500
- Number of LSTM units: 100
- Epochs: 13

As a conclusion of the results from all the tests above, Table 6 shows the results we achieved with different features for the model. We compare the best results in Table 5.3 which was 88.94% accuracy (with the SVM using the combination of unigram, bigram and trigram) with our result 91.04%, which is clearly higher with our model of deep learning. We performed many tests, but we covered only the tests that produced noticeable and significant results.

**Table 6:** Results of Our Model

	<b>Batch Size</b>	<b>Embedding</b>	<b>Max words</b>	<b>Max length</b>	<b>No. of LSTMs</b>	<b>Epoch</b>	<b>Result</b>
<b>Test 1</b>	100	Glove 6b-50D	1 M	2500	64	10	84.06
<b>Test 2</b>	100	Glove 6b-100D	1 M	2500	60	10	84.311
<b>Test 3</b>	100	Glove 6b-100D	1 M	2500	64	10	86.78
<b>Test 4</b>	100	Glove 6b-200	1 M	2500	64	10	88.83
<b>Test 5</b>	100	Glove 6b-300D	1 M	2500	64	10	89.36
<b>Test 6</b>	100	Glove 6b-300D	1 M	2500	60	10	89.72
<b>Test 7</b>	100	Glove 6b-300D	1 M	2500	64	10	86.30
<b>Test 8</b>	100	Word2vec 100b-300	1 M	2500	60	10	87.41
<b>Test 9</b>	100	Word2vec 100b-300	1 M	2500	100	10	90.03
<b>Test 10</b>	100	Glove 42-300D	1 M	2500	64	10	90.34
<b>Test 11</b>	100	Glove 42-300D	1 M	2500	100	13	91.04

## **CHAPTER SIX**

### **Conclusion**

#### **6.1 Conclusion**

Sentiment Analysis is a field that is still in development and has many enhancements that continuously expand it. With the revolution in AI and machine learning, the field gains more attention for scientists in the computer science field. This motivates them to experiment with different methods. The type of data can significantly change accuracy. This thesis described the methods used for the task of Sentiment Analysis. Regarding the literature review, the most widely used algorithms in the field are the supervised machine learning algorithms. Hence, we experimented with the Neural Network methods and in our case, the deep learning networks. We used the Recurrent Neural Network and compared it with the probabilistic classifiers and Support Vector Machine results described in [35]. Moreover, the work shared the same dataset. The results obtained from the paper were high and this required a high-quality model to surpass those results. In this work, the dataset was classified binarily for the IMDB dataset and each review was labeled as either positive or negative. We used a pre-trained word representation method (GLOVE) to weight the review for more accurate results. The word vectors were tested with different dimensions. During the research, we carried out many tunings and modifications to achieve the final result. The accuracy we achieved was 91.04% and we compared our result with the results from [35], which had 88.94% as the highest accuracy. Our method was better in test accuracy than the methods mentioned in the related work part. Our method surpassed the Naïve-Bayes (86.23%), Maximum Entropy (88.48%), Support Vector Machine (88.94%) and Stochastic Gradient Descent (85.11%) algorithms. Using the pre-trained word vectors improved the results and it reduced the time consumed on training and learning the pattern with better performance and greater accuracy.

## 6.2 Future Work

In the future, we will expand our work with a different model and layers, such as Recursive Neural Network or the Convolutional Neural Network. It might also provide a combination of many models. Moreover, the work will cover different datasets from many platforms to check the performance of the model and it will focus on social media datasets such as from *Twitter* or *Amazon*. Moreover, it will be tested with more properties than was used in the binary classification by estimating the results to many labels. Finally, we will endeavor to collect a large corpus and use it to build our own word vectors and use them in the model. Different word vectors will be used for training and they will contain the final developed word2vector model (Fast Text) which has been developed recently by Mikolov.



## REFERENCES

- [1]. Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1), 1-167.
- [2]. Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4), 1093-1113.
- [3]. Hailong, Z., Wenyan, G., & Bo, J. (2014, September). Machine learning and lexicon based methods for sentiment classification: A survey. In *Web Information System and Application Conference (WISA)*, 2014 11th (pp. 262-265). IEEE.
- [4]. Hu, M., & Liu, B. (2004, August). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 168-177). ACM.
- [5]. Hatzivassiloglou, V., & McKeown, K. R. (1997, July). Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics* (pp. 174-181). Association for Computational Linguistics.
- [6]. 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.
- [7]. Tang, B., Kay, S., & He, H. (2016). Toward optimal feature selection in naive Bayes for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 28(9), 2508-2521.
- [8]. Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.
- [9]. Maas, Andrew L., et al. "Learning word vectors for sentiment analysis." *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics, 2011.
- [10]. Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report*, Stanford, 1(12).
- [11]. Bütow, F., Schultze, F., & Strauch, L. *Semantic Search: Sentiment Analysis with Machine Learning Algorithms on German News*.
- [12]. Pak, A., & Paroubek, P. (2010, May). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In *LREc (Vol. 10, No. 2010)*.

- [13]. Gokulakrishnan, B., Priyanthan, P., Ragavan, T., Prasath, N., & Perera, A. (2012, December). Opinion mining and sentiment analysis on a twitter data stream. In *Advances in ICT for emerging regions (ICTer)*, 2012 International Conference on (pp. 182-188). IEEE.
- [14]. Hallsmar, F., & Palm, J. (2016). Multi-class sentiment classification on twitter using an emoji training heuristic.
- [15]. Salas-Zúrate, M. D. P., Medina-Moreira, J., Lagos-Ortiz, K., Luna-Aveiga, H., Rodreguez-Garcia, M. A., & Valencia- Garcia, R. (2017). *Sentiment Analysis on Tweets about Diabetes: An Aspect-Level Approach*. *Computational and mathematical methods in medicine*, 2017.
- [16]. Chiavetta, F., Bosco, G. L., & Pilato, G. (2016). A Lexicon-based Approach for Sentiment Classification of Amazon Books Reviews in Italian Language.
- [17]. Musto, C., Semeraro, G., & Polignano, M. (2014). A comparison of lexicon-based approaches for sentiment analysis of microblog posts. *Information Filtering and Retrieval*, 59.
- [18]. Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 38(11), 39-41.
- [19]. Dong, Z., Dong, Q., & Hao, C. (2010, August). Hownet and its computation of meaning. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations* (pp. 53-56). Association for Computational Linguistics.
- [20]. Kim, S. M., & Hovy, E. (2004, August). Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics* (p. 1367). Association for Computational Linguistics.
- [21]. Ding, X., Liu, B., & Yu, P. S. (2008, February). A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 international conference on web search and data mining* (pp.231-240). ACM.
- [22]. Thakkar, H., & Patel, D. (2015). Approaches for sentiment analysis on twitter: A state-of-art study. *arXiv preprint arXiv:1512.01043*.
- [23]. Rothfels, J., & Tibshirani, J. (2010). Unsupervised sentiment classification of English movie reviews using automatic selection of positive and negative sentiment items. CS224N-Final Project.
- [24]. Zagibalov, T., & Carroll, J. (2008, August). Automatic seed word selection for unsupervised sentiment classification of Chinese text. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1* (pp. 1073-1080). Association for Computational Linguistics.
- [25]. Sahni, Tapan, et al. "Efficient Twitter sentiment classification using subjective distant supervision." *Communication Systems and Networks (COMSNETS)*, 2017 9th International Conference on. IEEE, 2017.

- [26]. Gautam, G., & Yadav, D. (2014, August). Sentiment analysis of twitter data using machine learning approaches and semantic analysis. In Contemporary computing (IC3), 2014 seventh international conference on (pp. 437-442). IEEE.
- [27]. Pang, B., Lee, L., & Vaithyanathan, S. (2002, July). Thumbs up: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10 (pp. 79-86). Association for Computational Linguistics.
- [28]. Chikersal, P., Poria, S., & Cambria, E. (2015, June). SeNTU: sentiment analysis of tweets by combining a rule-based classifier with supervised learning. In Proceedings of the International Workshop on Semantic Evaluation, SemEval (pp. 647-651).
- [29]. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval (Vol. 1, No. 1, p. 496). Cambridge: Cambridge university press.
- [30]. Severyn, A., & Moschitti, A. (2015, August). Twitter sentiment analysis with deep Convolutional neural networks. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 959-962). ACM.
- [31]. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of words and phrases and their compositionality. In Advances in neural Information processing systems (pp. 3111-3119).
- [32]. Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188.
- [33]. Liu, P., Qiu, X., & Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. arXiv preprint arXiv:1605.05101.
- [34]. Aggarwal, C. C., & Zhai, C. (2012). A survey of text classification algorithms. In mining text data (pp. 163-222). Springer US.
- [35]. Tripathy, Abinash, Ankit Agrawal, and Santanu Kumar Rath. "Classification of sentiment reviews using n-gram machine learning approach." *Expert Systems with Applications*.57 (2016): 117-126.
- [36]. Garreta, R., & Moncecchi, G. (2013). Learning scikit-learn: Machine Learning in Python. Packt Publishing Ltd.
- [37]. McCallum, A., Nigam, K. et al. (1998). A comparison of event models for naive bayes text classification. In AAI-98 workshop on learning for text categorization (pp. 41-48). Citeseer volume 752.
- [38]. Nigam, K., Lafferty, J., & McCallum, A. (1999). Using maximum entropy for text Classification. In IJCAI-99 workshop on machine learning for information filtering (pp. 61-67).volume 1.

- [39]. Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade* (pp. 421–436). Springer.
- [40]. Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification.
- [41]. Raychaudhuri S, Chang JT, Sutphin PD, and Altman RB Associating genes with Gene Ontology codes using a maximum entropy analysis of biomedical literature. *Genome Research* 12:203-214, 2002.
- [42]. Berger A., *A Brief Maximum Entropy Tutorial*, 1997.
- [43]. Darroch, John N., and Douglas Ratcliff. "Generalized iterative scaling for log-linear models." *The annals of mathematical statistics* (1972): 1470-1480.
- [44]. Günther, Tobias, and Lenz Furrer. "GU-MLT-LT: Sentiment analysis of short messages using linguistic features and stochastic gradient descent." *Second Joint Conference on Lexical and Computational Semantics (\* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Vol. 2. 2013.
- [45]. Jadav, Bhumika M., and Vimalkumar B. Vaghela. "Sentiment analysis using support vector machine based on feature selection and semantic analysis." *International Journal of Computer Applications* 146.13 (2016).
- [46]. Azzouni, Abdelhadi, and Guy Pujolle. "A long short-term memory recurrent neural network framework for network traffic matrix prediction." *arXiv preprint arXiv: 1705.05690* (2017).
- [47]. <https://tensorflow.org/>
- [48]. <https://keras.io/>
- [49]. Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.
- [50]. <https://dumps.wikimedia.org/>
- [51]. <https://catalog.ldc.upenn.edu/LDC2011T07>