



TAMPEREEN TEKNILLINEN YLIOPISTO  
TAMPERE UNIVERSITY OF TECHNOLOGY

MUHAMMAD ADNAN MUSHTAQ  
MULTILINGUAL SENTIMENT ANALYSIS AS PRODUCT  
REPUTATION INSIGHT  
Master of Science Thesis

Examiner: University Lecturer  
Timo Aaltonen  
Examiner and topic approved on  
31st May 2017

## ABSTRACT

### **MUHAMMAD ADNAN MUSHTAQ: MULTILINGUAL SENTIMENT ANALYSIS AS PRODUCT REPUTATION INSIGHT**

TAMPERE UNIVERSITY OF TECHNOLOGY

Master of Science Thesis, 49 pages

November 20

Master's Degree Programme in Information Technology

Major: Data Engineering

Examiner: University Lecturer Timo Aaltonen

**Keywords:** Deep Learning, NLP, Sentiment Analysis, LSTM, RNN, text mining, classification, machine learning, Embedding, Activation Function, Hyper-parameter.

A vast amount of textual data is continually being generated every day and available across the internet. By utilizing this 'big data' we can deduct meaningful information from its contents. One of the major tasks of Natural Language Processing is sentiment analysis. In recent years sentiment analysis has gain much attention but this is an extremely challenging task due to the complexity of human language. Our day-to-day life has always been influenced by what people think. Behaviour of customer can be represented as sequential data describing the interest and views of customer about products. Earlier approaches like *Lexicon-Based* methods weren't able to extract deep insight from text data as those approaches were based on finding the density of negation, Irrealis-blocking.

To address this task deep learning has become popular method. LSTM (Long short-term memory) model has been used which is a modified version of RNN (Recurrent Neural Networks). Recurrent Neural Networks has ability handle sequential data very effectively and without performing any feature engineering it can learn directly from low-level features. Instead of exploring LSTMs abilities and capabilities, main focus was to learn how embedding can help us to understand user expectations from text. Proper pre-processing for data has been implemented. Informal language, contextualization, bad grammatical structure, misspellings are additional complicating factors. Reviews are analysed as binary classification task, after processing reviews are classified as either *negative or positive*. Features for training and testing the deep learning model were retrieved by using new method called '*word-vector*'. Moreover, effect of sentence length has also been investigated. Sentiment analysis for short sentences becomes difficult because of lack of contextual information. Multiple hidden layers have been used in the architecture. Dropout, Normalization and Parametric Rectified Linear Unit (PReLU) technology has been used to generalize and improve the accuracy of model. Also, the impact of various hyper-parameter has analysed. Different neural network configurations are evaluated. The performance of model is discussed with respect to the input data and model configuration.

## PREFACE

The research work presented in this thesis has been conducted at Tecnotree Corporation Tampere, Finland. This thesis is a culmination of my studies at Master's in Data Engineering. I finally carried out this project but it would have been very difficult without support and help of friends and people in my surroundings.

I can't thank you enough to my supervisors Ari Aalto (Office colleague), Timo Aaltonen for their support and excellent guidance throughout my work. I am also thankful to other respondents, without their cooperation it was a bit hard to conduct this research, especially Arsalan Ahmed and Shahbaz Ahmed.

Again, I sincerely express my gratitude towards my teacher Timo Aaltonen, for his advice, guidance for thesis writing. Finally, I am truly blessed to have a family which I have and you guys are just another reason why I will.

Tampere, 25.8.2017

Muhammad Adnan Mushtaq

## CONTENTS

1.	INTRODUCTION .....	1
1.1	Problem Statement .....	2
1.2	Aim and Objectives .....	3
2.	DATASET AND FRAMEWORK .....	4
2.1	Framework .....	4
2.2	Dataset .....	4
2.2.1	IMDB Dataset .....	5
2.2.2	Amazon Dataset .....	5
3.	RELATED WORK .....	6
3.1	Sentiment Analysis Difficulties .....	6
3.2	Sentimental Analysis Classification .....	7
3.2.1	Classification Levels .....	8
3.2.2	Classification Techniques .....	10
3.3	Perceptron .....	12
3.4	Neural Network .....	13
3.4.1	Forward Propagation .....	14
3.4.2	Back propagation .....	14
3.4.3	Activation function .....	14
3.5	Recurrent Neural Networks .....	16
4.	PROPOSED METHOD .....	18
4.1	Text pre-processing: .....	18
4.1.1	Tokenization .....	18
4.1.2	HTML Tag Removal .....	20
4.1.3	Emoticon Removal .....	20
4.1.4	Stopwords Removal .....	20
4.1.5	Lemmatization .....	21
4.1.6	Spelling Correction .....	22
4.2	Sequential Data .....	23
4.3	Word Vectors .....	24
4.3.1	Keras Embedding Layer .....	26
4.3.2	Word2Vec Embedding .....	26
4.4	Model Architecture .....	27
4.4.1	LSTM Layer .....	27
4.4.2	Dropout Layer .....	28
4.4.3	SpatialDropout1D .....	29
4.4.4	Batch Normalization .....	29
4.4.5	Dense Layer .....	30
4.4.6	Network Topology .....	31
5.	EXPERIMENTATION AND RESULTS .....	34
5.1	Hyper-parameter Optimization .....	34
5.1.1	Embedding dimension: .....	34
5.1.2	Number of Neuron: .....	34
5.1.3	Batch Size: .....	35
5.1.4	Dropout-Rate: .....	36
5.2	Results: .....	37
5.2.1	Performance on UMICH Dataset: .....	37
5.2.2	Performance on IMDB Dataset: .....	39
5.2.3	Performance on Amazon Dataset: .....	42

5.2.4 Resources Consumed ..... 45

6. CONCLUSIONS..... 46

7. REFERENCES..... 47

## LIST OF FIGURES

Figure 1.	Importance of sentiment analysis since 2004.....	2
Figure 2.	AWS sample data format.....	5
Figure 3	Sentiment classification techniques.....	8
Figure 4	An approach to convert non-grammatical words or phrases.....	10
Figure 5	Internal structure of an artificial neuron.....	12
Figure 6	Connection between layers in neural network.....	13
Figure 7	Weight assigning in forward propagation.....	14
Figure 8	Step function calculation based on threshold.....	15
Figure 9	Sigmoid function graph representation.....	16
Figure 10	Graph representation of ReLu function.....	16
Figure 11	RNN Looping.....	17
Figure 12	Anatomy of pre-processing tasks for text data.....	18
Figure 13	List of common emoji's.....	20
Figure 14	List of stopwords.....	21
Figure 15	Inheritance of StemmerI interface.....	22
Figure 16	Word vector approach example.....	24
Figure 17	1-to-N encoding.....	25
Figure 18	Repeating modules in Recurrent Neural Network.....	27
Figure 19	LSTM module contains four interacting layers.....	27
Figure 20	Network before and after Dropout.....	28
Figure 21	Dropout effect on training of network.....	29
Figure 22	Performance with and without batch normalization.....	30
Figure 23	Example of Dense layer in Feedforward neural network.....	31
Figure 24	Model network topology.....	32
Figure 25	Gradient descent calculation with different batch-size value.....	35
Figure 26	Probability ratio of dropping out neurons.....	36
Figure 27	No. of positive and negative reviews in UMICH dataset.....	37
Figure 28	Insight of UMICH dataset.....	38
Figure 29	Model accuracy on UMICH dataset.....	38
Figure 30	Error (loss) rate on UMICH dataset.....	39
Figure 31	No. of positive and negative reviews in IMDB dataset.....	40
Figure 32	Insight of IMDB dataset.....	40
Figure 33	Model accuracy on IMDB dataset.....	41
Figure 34	Error (loss) rate on IMDB dataset.....	41
Figure 35	No. of positive and negative reviews in Amazon dataset.....	43
Figure 36	Distribution of amazon data based on length.....	43
Figure 37	Model accuracy on Amazon dataset.....	43
Figure 38	Error (loss) rate on Amazon dataset.....	44

## List of Tables

Table 1.	Model prediction on UMICH dataset.....	39
Table 2.	Model prediction on IMDB dataset.....	42
Table 3	Predicted values by model trained with amazon dataset.....	44
Table 4.	Analyses of models among different dataset.....	44
Table 5.	Resources consumed by model on different datasets.....	45

## LIST OF ABBREVIATIONS

NLP	Natural Language Processing
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
DBN	Deep Belief Networks
CNN	Convolutional Neural Networks
GPU	Graphic Processing Unit
SNN	Shallow Neural Network
NN	Neural Network
DL	Deep Learning
IMDB	Internet Movie Database
AWS	Amazon Web Service
SA	Sentiment Analysis
NLTK	Natural Language Toolkit
HTML	Hypertext Mark-up Language
TF-IDF	Term frequency–inverse document frequency
BoW	Bag of Words
GloVe	Global Vectors for Word Representation
CBOW	Continuous Bag of Words
UMICH	University of Michigan
ReLu	Rectified Linear Unit

# 1. INTRODUCTION

Today extensive datasets are accessible on-line, holding text data or numerical. It has been the major focus for many practitioners and researchers to apply reasonable approaches and techniques and extract useful information from those datasets. Wide range of techniques have been proposed and tested to retrieve information during this time [1]. In addition to text mining and data mining, lately interest for non-topical text analysis have increased drastically and *sentiment analysis* is part of them.

Sentiment analysis is a process of analysing the given text in order to find out the emotions in it. Sentiment analysis is about “Text analysis, Information Extraction and Natural Language Processing are kind of tasks which aim towards getting the writer’s feelings expressed in negative or positive comments by analysing sentences or documents” defined by Subhabrata Mukherjee [2]. In simple words, opinion mining is a process of detecting the sentiment of the writer concerning a particular topic. It is a blend of techniques and strategies about distinguishing and detecting subjective information from a text such as opinions and attitudes [3]. Usually, it has been about opinion polarity to find out whether someone has negative, positive or neutral opinion about something [4].

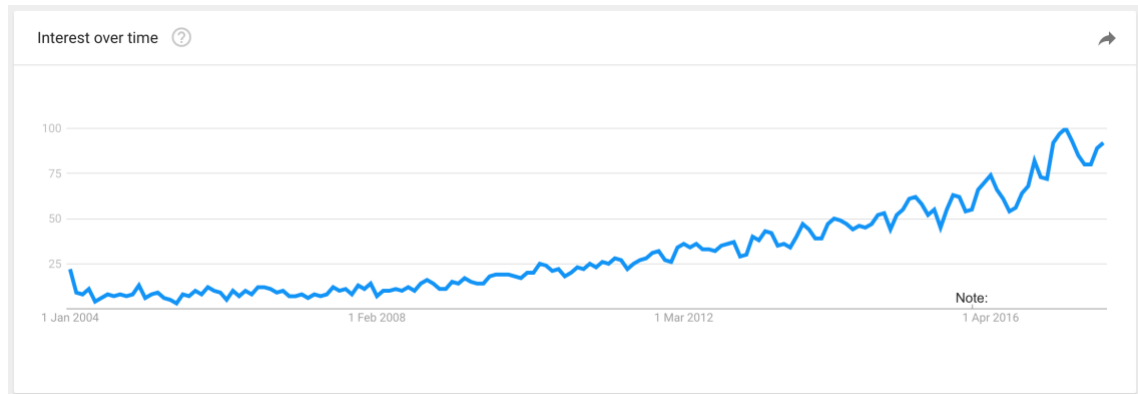
Internet is a huge source of information for every individual. For instance, investors want to be updated with financial news particularly associated with their investment. Organizations are looking for the news about competitors, suppliers and customers feedback. This is a cyclic chain, likewise customers have interest in reviews of other customers about the products they are looking for. Researcher made an excessive amount of effort to identify the impacts of this technique on customer insight, trend and financial world. Numerous applications of sentiment analysis came up in multiple domains like sentiment analysis of products reviews, financial news and healthcare [5]. It additionally offers them a superior picture of how they stack up against their competitors.

Technically, sentiment analysis is a unique blend of artificial intelligence and machine learning, allowing organizations to use advanced tools to choose useful and reasonable moves that attract consumers toward their services and products. In order to retain customers, competitors have to track and monitor the interest of customers. Especially, not towards their own products and brands only but also towards their competitors.

In figure 1 Google trends shows that “Sentiment Analysis” has been gaining steady traction over the past 13 years. This also shows wide usage and importance of sentiment



analysis.



**Figure 1. Importance of sentiment analysis since 2004.**

Machine learning have seen rapid change in previous two years with significant breakthroughs in deep-learning approaches. *Deep neural networks* enlivened by the human brain architecture and with enough processing power these models have been shown unbelievable results on many complex problems including Natural Language Processing tasks, even without having excessive domain knowledge. Out there many neural networks are available with their classic abilities like *Deep Belief Networks* (DBN) with fast inferencing of the model parameters, *Convolutional neural networks* (CNN), and *Recurrent neural network* (RNN). In this work, I will work with LSTM (Long Short-Term Memory). LSTM networks are a type of RNN that uses special units in addition to standard units [6].

## 1.1 Problem Statement

Research in sentiment analysis has targeted on two issues mainly, type detection of text whether it's objective or subjective, and if subjective then further is it negative or positive. Two main approaches were used:

1. Simple lexical analysis.
2. Classifications through supervised and unsupervised ML approach.

Classification of text written in natural language into a negative or positive is sometimes so complicated that even human doesn't agree on the classification assigned to text. Interpretation by an individual is not the same as others, and this can be influenced by cultural factors and person's experience. If there are other problems like shorter the text, and the worse written, the task becomes more difficult.

Despite of research done so far, challenge of this field is to enhance the machine's capability to understand texts similarly as human readers. For this purpose, we can take advantage of massive amount of opinions expressed on the internet and deep-learning.

## 1.2 Aim and Objectives

Two main objectives:

- To explore the key ways for the improvement of sentiment analysis performance and
- A model which can predict sentiment for multiple languages at same time (At least two, e.g. English and French/Spanish).

For this purpose, Long short-term memory model (explained in section 4.4.1) is a good choice which is a type of Recurrent neural network (Explained in section 3.5). Long Short-Term Memory is a redesign of the RNN architecture around special ‘memory cell’ units [7]. Thus, NN require large amount of data is for training purpose. However, currently available supervised data is not enough so far, in that case weakly labelled data will be used to train and test the model.

## 2. DATASET AND FRAMEWORK

This chapter contains the information about the framework and dataset, used to train Deep Learning models and the text representations, and also for experimentation in this thesis.

### 2.1 Framework

For this sentiment analysis task, Keras<sup>1</sup> has been used for modelling the *DL*<sup>2</sup> (deep learning) models. Keras is a programming framework for deep learning and its written in Python programming language. It is a minimalistic library with a focus on fast experimentation and simplifies the process of building applications based on deep learning. Keras can run on top of Theano or TensorFlow, both of them allows running computations over GPU's. Theano is a library for fast numerical computation. It's a compiler for mathematical computation in Python and was developed by MILA group at University of Montreal, Canada. TensorFlow was created by Google to replace Theano. These two libraries are quite similar but TensorFlow has tools to support *Reinforcement learning*. Reinforcement learning is a type of machine learning which allow machines to automatically determine the ideal behaviour in a specific context.

*Scikit-learn* is another library which provides a range of unsupervised and supervised learning algorithms via a consistent interface in Python. But keras library is handier than scikit-Learn. It gives freedom to define our own designed machine learning models, rather than pre-defined ones. We can run Keras on top of TensorFlow. As Google is putting efforts in making TensorFlow the fastest, so this way we can get those benefits. Currently TensorFlow is a scalable (deep learning) engine in the industry.

Combination of Keras and TensorFlow<sup>3</sup> has been used for sentiment analysis task. Another package Gensim<sup>4</sup> which has been used for word vector handling. Gensim is a Python library which is designed to extract semantic topics from documents. Algorithms used in gensim library are unsupervised and this library is designed to process unstructured, raw text data. Gensim is an exceptionally optimized, yet additionally very specific, library for doing tasks related to text data. It offers a simple, surprisingly efficient AI-approach to handle raw texts and it is based on *SNN*<sup>5</sup> (Shallow Neural Network).

### 2.2 Dataset

There is a lack of annotated datasets for the problem of opinion mining. It's a novel problem and that's why we need benchmark dataset. Additionally, the available datasets do not use a well-formulated guideline and definition of suggestions [8].

---

<sup>1</sup> Keras: <http://keras.io/>

<sup>2</sup> A branch of machine learning based on the structure and function of brain.

<sup>3</sup> TensorFlow: <https://www.tensorflow.org/>

<sup>4</sup> Gensim: <https://radimrehurek.com/gensim/>

<sup>5</sup> Type of neural network has only one hidden layer.

## 2.2.1 IMDB Dataset

This dataset contains a collection of 50,000 polar movie reviews. Labelled as either negative or positive. Negative reviews hold fewer stars than five stars whereas positive ones were rated with more than six stars. This IMDB data<sup>6</sup> has been taken from Stanford University. Researchers in Stanford University collected IMDB data and performed sentiment analysis on that. They achieved 88.89% accuracy. Now the dataset is properly divided, 25,000 for training and 25,000 for testing [9].

## 2.2.2 Amazon Dataset

AWS (Amazon Web Services) hosts a number of public datasets. The datasets are available for free. Product reviews dataset<sup>7</sup> has been used for this thesis, including ~ 143 million reviews spanning May 1997 – May 2014. Reviews contain information about user, product type, ratings, text review etc. Review by each customer was handled as JSON object in dataset. Figure 2 shows a sample review taken from Amazon product review dataset. Sentiment analysis is concerned with reviews so, ‘review/text’ part was extracted from the dataset.

Data Format:

```

product/productId: B00776WASK
product/title: Rock Rhythm & Doo Wop: Greatest Early
Rock
product/price: unknown
review/userId: A1RSDE930N6RSZF
review/profileName: Joseph M. Joseph
review/helpfulness: 7/9
review/score: 3.0
review/time: 1241502400
review/summary: Pittsburgh - Home of the OLDIES
review/text: I have all of the doo wop DVD's and this
one is as good or better than the 1st ones. Remember
once these performers are gone, we'll never get to see
them again. Rhino did an excellent job and if you like
or love doo wop and Rock and Roll you'll LOVE this DVD!!

```

*Figure 2. AWS sample data format*

<sup>6</sup> IMDB: <http://ai.stanford.edu/~amaas/data/sentiment/>

<sup>7</sup> AWS: <https://snap.stanford.edu/data/web-Amazon.html>

### 3. RELATED WORK

There has been considerable research done and still going on sentiment analysis subject. Sentiment analysis became part of research at the beginning of 20<sup>th</sup> century and in 1990, text subjectivity analysis was performed by computational linguistic community [1]. Most of the sentiment analysers used to work by choosing one or hybrid of following approaches.

- Using Vocabulary:  
Worked by choosing the important keywords (usually adjectives and verbs) along with modifiers. For example, negative words.
- Machine Learning approach:  
Treat sentiment analysis as classification problem, extract features. Train model to determine sentiment.
- Using Rules:  
Look for presence of specific words in sentence and define rules based on those words and categorize sentences.

In following sections, old approaches for classification of sentiment analysis and difficulties in sentiment analysis has explained.

#### 3.1 Sentiment Analysis Difficulties

Research demonstrates that the task of sentiment analysis is more tough than conventional topic based classification of text, regardless of the fact that we don't have much classes in SA than classes in topic based classification [10]. In this task, usually classification assigned to the text are generally positive or negative. There can be some different binary classes also or multivalued classification. For example, *neutral, negative or positive*, yet those classes are not as much as in topic based classification. Topic based classification is a bit easier than sentiment analysis because this can be achieved with the use of keywords this could be a reason. On the other hand, this technique doesn't perform well with sentiment analysis [11].

Classification in sentiment analysis is a subjective method but there could be variations in opinions if there are number of observers to test. Interpreting the state of mind of a subject may differ person to person and if someone has only 140 characters or less to express something then its significantly hard to determine the mood [12].

The study of sentiment classification and subjectivity classification is required to perform sentiment analysis properly. In subjectivity classification, it is identified that whether or not provided text data contains opinionated information or factual information. Similarly, sentiment analysis is a process to classify an opinion into negative or positive. In reality, if we consider a product review then it requires in depth analysis of assigned classification because the manufacturer of product is interested to know the details of opinion, for

example owner wants to know what features of a product have been criticised or praised.

Following is the example of review posted by a user on a pair of Shoes:

- 1) I have rated those shoes 4 stars because such a cool pair of shoes, but had a few problems.
- 2) Order a half size down from your regular fit.
- 3) Uncomfortable in a few places, but overall not too bad.
- 4) Arrived 4 days late (DHL's fault), and had a small beige stain next to laces, as well as a black scuff on the white sole.
- 5) If I was a collector I would have wanted a refund, but I couldn't be bothered to send them back as I had already wait long enough.
- 6) I would recommend those shoes as they are light weight.

In this situation, the main problem is that what exactly we want to extract from this review. It's easy noticeable that there are variety of opinions provided in this example, sentence (6) express a positive review on shoes while sentence (1) and (3) could be positive or negative. The remaining sentences (4) and (5) are inclined toward negative opinion. Number of opinions in the sentence by user have some targets on which user expressed views. Sentence (4) and (6) are based on the feature of shoes such as quality, shape, fitness whereas sentence (5) is about delivery, nothing related to product. This review example helps to understand the difficulties and challenges in opinion mining or sentiment analysis are directly proportional to deep understanding and require immense data analysis to have precisely analysed opinion. There can be other factors which increases the difficulty in sentiment analysis for e.g. text expressed with irony, sarcasm or negation.

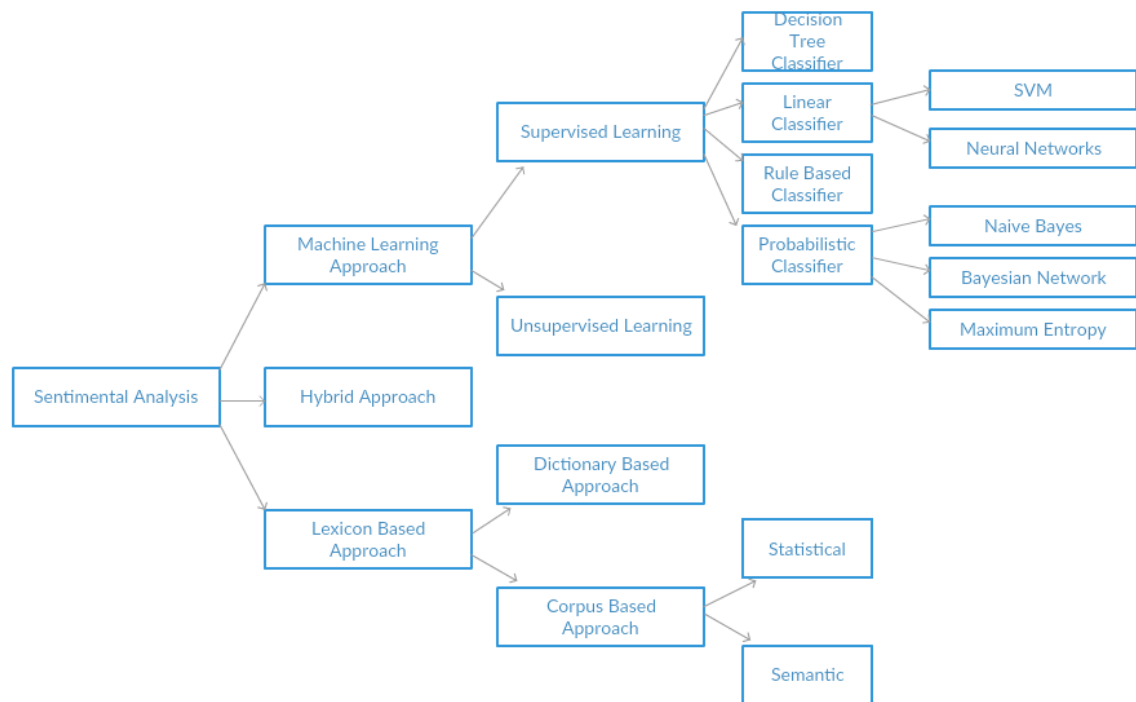
### 3.2 Sentimental Analysis Classification

Sentimental analysis is classified into multiple sentimental classification techniques. Two of them are popular, first one is machine learning approach and other is lexicon based approach. There is also third classification technique which is known as hybrid approach (show in figure 3) and it uses of the above-mentioned classifications to optimize the solution [13]. Following is the introduction of mentioned approaches.

- **Machine Learning Approach:** This approach merely depends on text analysis and classification. Text analysis is mainly used for business decision making, for which it require text processing. Initially, it requires some collection of data to train a model, which later serve and help in prediction of new set of data without any sort of labels. Model predicts the unlabelled records by predicting their labelling class. Classes are classified as positive, negative or neutral. Machine learning approach is further divided into following to learning methods.
  1. Supervised Learning
  2. Unsupervised Learning

- **Lexicon Approach:** In this approach, there is a need to define the dictionary or collection of words and phrases with their synonyms and antonyms. Most common approaches used in lexicon for the collection of words are following two.
  1. The Dictionary Based
  2. The Corpus Based
- **Hybrid Approach:** It is most efficient and optimized approach among all, it can identify and detect the emotions from a text. Support Vector Machine Algorithm, which works on a technique to find the best available linear separator between the classes, is required to achieve the goal of hybrid approach.

Sentimental analysis can also be classified in terms of levels and ratings. The classification of these sentiments is based on opinion polarities (positive, negative or neutral) [14]. Figure 3 illustrates the hierarchy and tree structure of classification as discussed earlier.



**Figure 3** *Sentiment classification techniques*

### 3.2.1 Classification Levels

This sub chapter explains about the sentimental analysis classification levels. Following are the levels in which sentimental analysis is classified.

#### 1. Sentence Level

It is used as first filter of analysis. In this level of classification, every single sentence is taken under consideration to analyse and express the opinion [15]. Sentence level works on the assumption that there must be single sentiment against one sentence.

This presumption is not necessary for all the sentences in the given collection or document. The most important thing in this classification is to discriminate between biased and non-subjective sentences. Non-subjective sentences provide no information in decision making. Contrarily to this, subjective sentiment provides opinion and detection of those sentences which contains some facts [16]. Sentence level classification provides help to prevent misleading and selecting irrelevant data or sentences. As a result, it is used to increase the efficiency and performance of sentence level sentiment classifier. It is preferred to use when there is need to have more than one opinion in one document. It also provides support to treat sentences differently for special classification. Best scenario of using Sentence Level classification is on conditional and comparative sentences. It is assumed that there is no single strategy available to different sentences or whole text of all the types. In order to improve the accuracy, using combination of different strategies is preferred. It is also preferred to rate opinions in terms of positive or negative opinion, not in terms of good or bad opinions.

## **2. Document Level**

Main objective of this classification is to find out, either whole document has positive or negative opinion. This method considers whole document as a single entity and that is why is not suitable in situation where evaluation of more than one entity requires.

This level is sentiment analysis faces a lot of criticism because of its way of working. As, it is unrealistic due to the fact that there may be many possible opinions in the text. On the other hand, it is useful in situations where some reviews or final statement about the product are required. Other use-case of document level sentiment is scenario news carries some positive or negative opinions and these opinions reflect in terms of buy or sell signals.

## **3. User Level**

This is not a famous or popular level of sentimental analysis but researchers have defined some use-cases for this sort of analysis in a situation where user wants to observe user's network based on the behaviour of the neighbour users [17, 18].

## **4. Aspect Level**

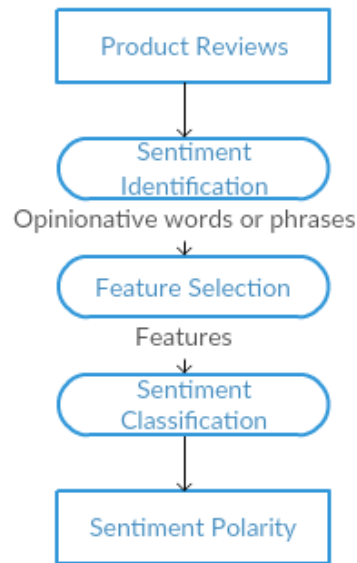
It is also known as feature level or phrase level analysis. It is different from other classification levels in term of method of evaluation. It analyses in such a way that first it finds the target and then discover its opinion. Other the other hand, other classification levels focus on languages units such as sentences, documents and paragraphs. The aim of this classification is not to find the opinion of entities but also their different aspects.

This analysis can be achieved by differentiating polar phrases and defining their sentiments from other [19, 20]. In finance, this level of sentiment is used to find the relationship between detected polar words and other variables, example of which is firm earnings and stock prices [21].

All the models built for product analysis and mining of customer opinions about certain product feature is based on aspect level sentiment [22, 23]. In general, all the words or phrases in aspect level sentiment directed to specified topic or an object.



Figure 4 depicts the process of extracting aspects of reviewed products based on the opinions of customers. Feature selection process is performed to extract feature and based on those feature, trained models categorize the review.



**Figure 4** *An approach to convert non-grammatical words or phrases*

### 3.2.2 Classification Techniques

In sentiment analysis classification techniques are of core importance and most of work is also done on based on it. The main objective of these techniques is to separate the positive, negative or neutral opinions in document [24]. This sub-chapter mainly focuses on sub-division of machine learning approach and lexicon-based approach, which are discussed previously in chapter 3.2.

*Machine learning approach* is based on text classification, which is used for forecasting and business decision making by automating the processing text. This approach is divided into supervised and unsupervised learning. In supervised learning approach, initially models are trained using classifiers of document. These trained models or documents have some key features, which have topic related words. Supervised learning is further classified into following.

- **Decision Tree:** It is used for prediction and are used for classification as well. If record is given with unlabelled or unclassified class label, then compared with decision tree, which is traced from the root to node and find outs the prediction of class against submitted record [25]. Decision tree is popular because it does not require any configuration expertise. Some of the packages used for implementation of decision tree in text classification problems are *ID3* and *C5*.

- **Linear Classifiers:** These techniques are famous due to simplicity. The objective of this technique is to find out number of opinions in provided data and find their polarity by comparing them with list of pre-defined words. Weights are added against words in such a manner, a word with *most negative* opinion has lowest weight, on the other hand, word with *most positive* opinion have highest weight. Most popular type of Linear classifiers is Support Vector Machine (SVM) classifiers. [26]
- **Rule-Based Classifier:** It is same to some extent to technique of decision tree, these techniques are based on rules and feature space [27]. The main distinguish in term of that rule-based classifier allows overlap in the decision tree [26] whereas, decision tree classifier uses hierarchical approach. In this classifier, rules are generated based on different criteria's, such as support and confidence [28].
- **Probabilistic Classifier:** It is also said to be generative classifiers as it generates a model against each class [29]. It assumes that every class is a part of model. The most widely used probabilistic classifier is *Naive Bayer Classifier*, which is simplest to implement in any programming language due to the fact that it involves simple mathematics [30]. It works on the principle that each model consists of scattered set of words, frequency of existed words remains same but not the spot. Naive Bayes uses Bayes Theorem, which allows the label to find out the set of features.

Unsupervised learning is used in a situation when it is difficult to create a class-labelled document, which makes it more natural and general as compared to supervised learning. For that purpose, unsupervised learning is implemented on collected unlabelled documents. In case of document clustering analysis, this learning approach is mainly used because it not only relies on already defined class labelled training documents. It is different from supervised learning in such a way it learns by observation and pre-defined models are not submitted to solution. [31]

Another unsupervised approach is *Lexicon-based*, which uses dictionary. This dictionary consists of list words and phrases, mainly synonyms and antonyms with opinions. Most automated and accepted sentiment word list used for *Lexicon-based approach* are following.

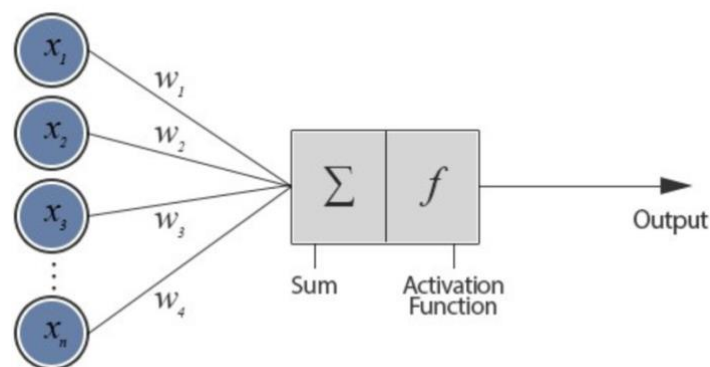
- **Dictionary-Based Approach:** It's works on the basic principle that several small set of opinion words are collected manually together to transform it in large collection of text [32]. Every time new word is found, it is added into existing document and this cycle repeats until no unique remains. The major disadvantage of this approach is that it totally depends on large collection of data and it is not possible to enter almost each opinionated word manually created document [33].
- **Corpus-Based Approach:** Main use of this approach is in scenario where there is need to discover new sentiment word or text from domain of collection in list of already known opinion words and to generate new sentiment lexicon from other [12]. Downside of this approach is that it will work efficiently only in case when

collection of all the English or any language words are already present in pre-defined document [34]. It is further divided into statically and semantic approach.

There is also third type of sentimental analysis approach, it is known as *Hybrid Approach*. The basic working principle of this approach is to find out anticipation from a text with or without affecting associated words. In order to obtain effective identification, Support Vector Machine Algorithm are also used in this approach. Some of industries like HP, are using mixture of Machine Learning and Lexicon-Based approaches together to create hybrid based approach.

### 3.3 Perceptron

Mathematical representation of biological neuron is called perceptron. Actual neuron produces a signal only when the strength of a signal exceeds a certain threshold. Similarly, in perceptron each input is multiplied by a certain value called weight. Weight of multiple inputs is calculated to represent the strength of signal and an activation function is used to find out the output value [35].



**Figure 5** *Internal structure of an artificial neuron*

In figure 5 weights ( $w_1, w_2, \dots, n$ ) and inputs ( $x_1, x_2, \dots, n$ ) are real numbers and those numbers can be negative or positive. All input values are weighted individually then added and finally sent to activation function. There are multiple types of activation functions (explained in 3.4.3) but the simplest one is *step-function*. A step function works in a way that if the input is greater than a certain threshold it will produce 1 otherwise 0.

For example:

Input 1 ( $x_1$ ) = 0.7 and weight 1 ( $w_1$ ) = 0.6

Input 2 ( $x_2$ ) = 1.1 and weight 2 ( $w_2$ ) = 0.9

Threshold = 1.0

Calculation:

$$x_1 * w_1 + x_2 * w_2 \rightarrow (0.7 * 0.6) + (1.1 * 0.9) \rightarrow 1.41$$

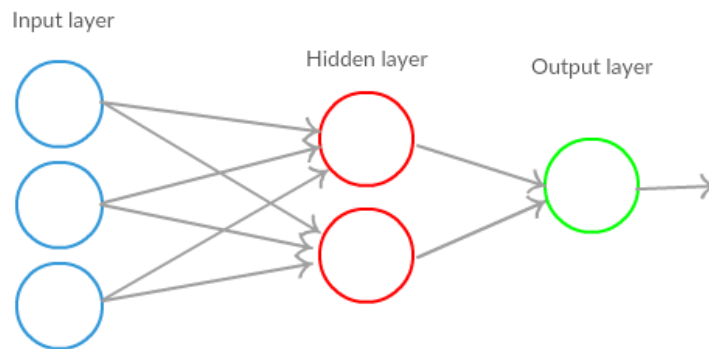
As the calculated value is greater than threshold so the output will be 1 that case.

### 3.4 Neural Network

In terms of computer science, a neural network is an artificial nervous system for receiving, processing and transmitting the information. Collection of neurons with *synapses*<sup>8</sup> which connects them is called a neural network. There are three different types of layers in a neural network:

- Input Layer:
  - Input fed to network through this layer.
- Hidden Layer:
  - This layer processes the input taken from input layer and there can be multiple hidden layers.
- Output Layer:
  - This layer produces the processed data.

Figure 6 illustrates the connection between those layers. Circle represents the neurons and the line connecting them represents synapses



**Figure 6** Connection between layers in neural network

#### Input layer:

Input layer presents a pattern to neural network. This layer only deals with input data. Each neuron in input layer should represent an independent variable that has some effect on the output of network [36].

#### Hidden layer:

Hidden layer is also a combination of neurons and also has activation function. This layer is also known as middle layer. The main job of hidden layer is to extract important features from data fed by previous layers or layer. There can be multiple hidden layers in network depends upon the complexity of problem. For example, if data can be separated linearly then there is no need to use the hidden layer as activation function can be implemented directly on input layer. If a problem need complex decisions then we can use more than one hidden layer. It's not sure that increasing the number of hidden layers will result in high accuracy. At some extent accuracy becomes constant or falls if an extra layer has been added. Number of neurons also effect the accuracy result. If number of neurons are less than complexity level of problem then there will be few neurons in the

<sup>8</sup> Connection between two neurons is called synapses.

network to detect the signal from complicated data. Similarly, if there are excessive amount of neuron used then *over-fitting* (explained in section 4.4.2) may occur.

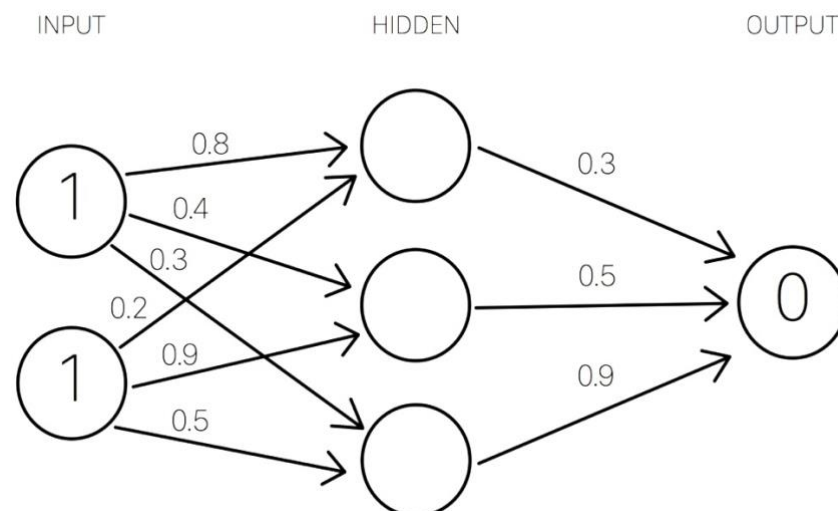
### Output Layer:

Output layer collects and produces the results in a way that it has been designed to produce. Typically output layer make predictions for classes.

Neural network training means calibrating the weights and calibration is achieved by repeating forward propagation and backward propagation.

### 3.4.1 Forward Propagation

In forward propagation set of weights are applied to the input data and output is calculated. This process can be repeated multiples time depends upon complexity of problem but for the first-time weights are selected randomly. Figure 7 represents a general process of forward propagation in which weights to synapses assigned randomly.



**Figure 7** Weight assigning in forward propagation

### 3.4.2 Back propagation

After assigning random weights in forward propagation an output is calculated and compared to actual expected result to get the error. To decrease the error, weights are propagated backward by taking derivative of error according to each weight and subtracting this value from the weight. Calculation for back propagation occur at each layer. Back propagation and forward propagation works together but to be precise forward propagation is a part of back propagation algorithm but it occurs before back propagation [37].

### 3.4.3 Activation function

Activation function is an important feature of neural network. Neuron should be activated or not is decided by activation function. It calculates the weighted sum of inputs and add bias. It's a non-linear transformation of input value. After transformation, this output is

sent to next layer. Without an activation function a neural network is just a linear regression model.

$$Y = \sum (\text{weight} * \text{input}) + \text{bias}$$

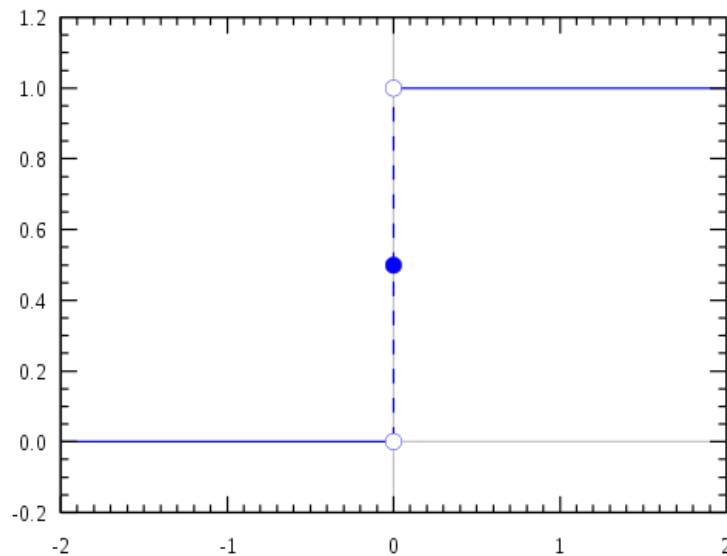
Equation No. 1

## Types of Activation function:

### 1. Step function

It's a threshold based activation function. Simple, if value of X is greater than a certain value mark it activated otherwise not.

Activation function  $Y = \text{"not"}$  if  $X < \text{threshold}$  else  $\text{"activated"}$

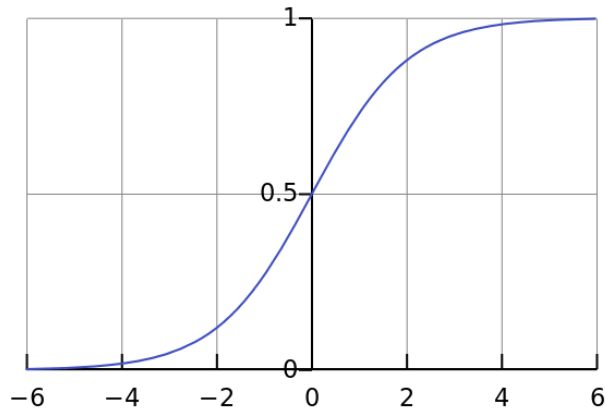


**Figure 8** Step function calculation based on threshold

As shown in Figure 8 [38] output is 1 when value is greater than 0 (zero is the threshold value) and outputs is 0 if less than threshold.

### 2. Sigmoid function

Sigmoid function is like a step function but its non-linear in nature. Its output ranges between 0 and 1 and has a S shaped curve. From figure 9, its easily noticeable that X values between -2 to 2, values of Y are very steep. It means that small change in X will result in significant change in values of Y.



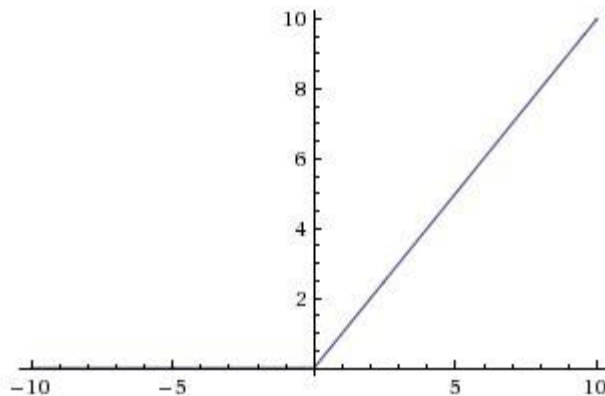
**Figure 9** Sigmoid function graph representation

### 3. ReLu Function

It's a very simple activation function. Suppose input is value  $X$  and if  $X$  is positive the output will be  $X$  otherwise 0. ReLu (rectified linear unit) function is:

$$\text{Function}(X) = \max(0, X)$$

In Figure 10 there is a straight line and it looks like it's a linear function but ReLu is non-linear in reality. The range of ReLu is  $[0, \text{infinity}]$ . Computationally ReLu is less expensive than other activation functions because it has simple mathematical operation.



**Figure 10** Graph representation of ReLu function.

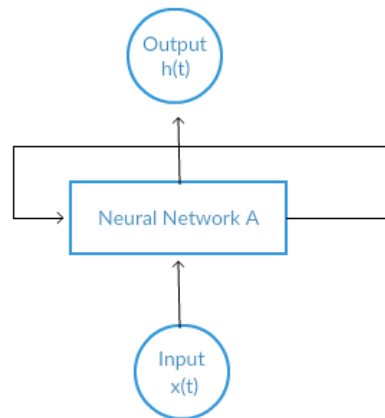
### 3.5 Recurrent Neural Networks

This Chapter explains the difference between human computation and memorizing power and neural networks. In addition to that, it also explains how Recurrent Neural Networks (RNNs) are better than traditional neural networks. It is a human nature that no one start thinking about the situation or problem from the very initial, every now and then. Humans try to sort out the efficient solution of a problem depending on their previous knowledge and understanding. On the other hand, machine neural networks lack this power of taking decision and analysing situations depending on the past information.

In traditional neural networks, it was difficult for machine to memorize the background information. This problem can be elaborated by an example of a movie, consider a machine needs to determine what sort of events could happen next at every scene. It was

nearly impossible for traditional neural networks to deal with such sort of situations. Contrarily to this, Recurrent Neural Networks can handle these issues. In the past few years, RNNs are used widely and almost eliminate use of traditional neural networks. They can be used to solve vast variety of problems, which include, language modelling, image captioning, speech recognition, translation and so on. [39]

Like control systems, Recurrent Neural Networks works on the principle of looping and chaining, which can be elaborated using Figure 11.



**Figure 11 RNN Looping**

From Figure 11, consider A is a small piece of neural network. This chunk needs some input value to start the computation and processing and result some output value, which are  $x_t$  and  $h_t$  in this case. Neural Network A, also forms a loop, which is basically a numerous copy of same network and allows knowledge and information sharing between next steps in the network. Hence, this feature of knowledge sharing distinguishes RNNs from other neural network techniques. Moreover, chain like approach of RNNs enables them to resemble with sequence and list. [40].



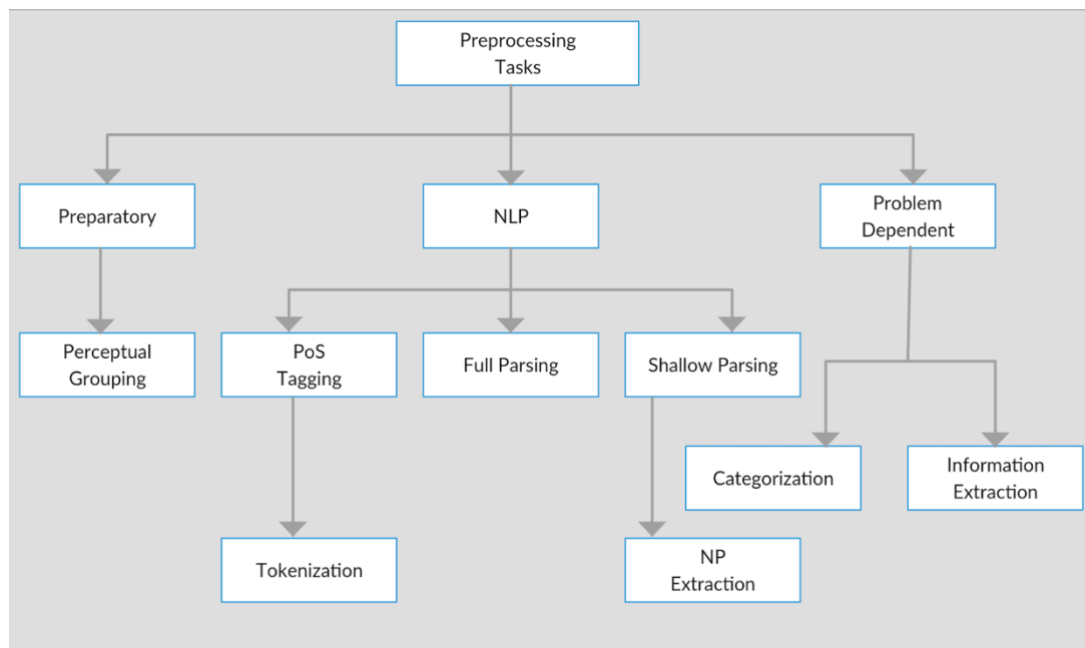
## 4. PROPOSED METHOD

This section will explain the techniques used for thesis.

### 4.1 Text pre-processing:

Majority of the text data encountered is in unstructured<sup>9</sup> format. This type of text appears at many places like emails, newspaper and social media. By unstructured, it implies that this content exists "in the wild" and has not been processed. For IMDB and Amazon Product reviews dataset some pre-processing required to feed data to neural network.

All the possible text pre-processing tasks are shown in figure 12 [11]. According to the requirement of task steps are chosen to perform pre-processing. For sentiment analysis task NLP approach is better to use than rest of approaches. By applying NLP approach multiple features can be extracted for example PoS tagging, tokenization.



*Figure 12 Anatomy of pre-processing tasks for text data.*

Following are the steps used in this thesis for text pre-processing.

#### 4.1.1 Tokenization

In tokenization, a continuous stream of character must be broken down and this can be done at different levels. A paragraph can be broken up into words or sentences. But in our case, we are splitting the unit of text into sentences. There can be different forms of

<sup>9</sup> Data which contains HTML tags, spelling mistake, Acronyms

this process depends upon the language we are dealing with. In-case of English, an effective and straightforward strategy for tokenization is to use punctuation and white space as token delimiters. This strategy is easy implement but it may not match with desired behaviour at some instances. In case of abbreviations and acronyms for example, U.S.A (an abbreviation for the United States of America) this will be tokenized as separate tokens U, S and A by losing the exact acronym meaning. To address this problem there are two approaches.

- Smart Tokenization, this approach avoids tokenizing the abbreviations and acronym.
- Handle the abbreviations and acronym prior pushing to tokenization or any other step.

For this simple tokenization task *NLTK*<sup>10</sup> tokenizer is a good choice. In this tokenizer *sent\_tokenize* function uses an instance of *PunktSentenceTokenizer* (algorithm) from the *nlk.tokenize.punkt* module. It's trained enough to work multiple European languages [41]. In the following code snippet, a string has passed to tokenizer function and tokenizer splitted every sentence in the string.

```
>>> from nltk.tokenize import sent_tokenize as st
// paragraph is the input data
>>> paragraph = "Its June and it hot already!. September was cold. Life is not that
much easy. But I like that"
// passing paragraph string to tokenizer
>>> st(paragraph)
// output after tokenization
["Its June and it hot already!", "September was cold.", "Life is not that much
easy.", "But I like that"]
```

Tokenization for languages other than English:

For any other language, we need to load *pickle*<sup>11</sup> for that language. Here's an example for French Language. In the following snippet, first we load the pickle for French language then passing string to tokenizer function.

```
// loading French dictionary
>>> french_tokenizer =
    nltk.data.load('tokenizers/punkt/PY3/ french.pickle')
//passing string to tokenizer
>>> french_tokenizer.tokenize(' Salut comment ca va. Comment ça se passe')

['Haila ammigo.', 'Estioy baien.']
```

<sup>10</sup> suit of programs and libraries to perform NLP tasks.

<sup>11</sup> Dictionary for a language

### 4.1.2 HTML Tag Removal

Text on the web is mostly in the form of HTML documents and there is a possibility that a paragraph or corpus, may contain some HTML tags which can affect the result of analysis. Before proceeding further, in a small test for checking the HTML tags. I have seen that our text data contains a lot of HTML tags such as `<pre>`, `<br/>`. *BeautifulSoup*<sup>12</sup> is a Python library which can be used to remove HTML tags. It's a library for pulling data out of XML and HTML files. Pass raw text data to library and call `get_text()` function to retrieve clean text as show in following code snippet.

```
>>> text_cleaned = BeautifulSoup(text_raw).get_text()
```

### 4.1.3 Emoticon Removal

In our modern textual inputs emoticons have risen increasingly important because people use this feature excessively to present their attitude [42]. Emoticon can be helpful to evaluate the polarity of positive and negative sentiment but in our scenario, we have ignored this feature. For the removal of emoticons, I have used Python Regular expression. Figure 13 shows some of the commonly used emoticons.



*Figure 13 List of common emoji's*

### 4.1.4 Stopwords Removal

Stopwords are words which do not contain important significance in a sentence. For most of the text analysis tasks, in order to speed up the task or save the storage it is helpful to remove stopwords as they appear a lot of times in corpus. Those common words generally

<sup>12</sup> <http://www.crummy.com/software/BeautifulSoup/>

do not contain useful information also don't contribute to the sentence meaning [43]. At least in natural processing tasks, these words are *that, this, there*. Also search engines skip the stopwords from query so that they can save space in index. Figure 14 shows list of some common stopwords.

Without losing the information we can remove the stopwords, because for most of the algorithms and text mining tasks such words have very low impact on the results.

```
hers, between, yourself, but, again, there, about,
once, during, out, very, having, with, they, own,
an, be, some, for, do, its, yours, such, into, of,
most, itself, other, off, is, s, am, or, who, as,
from, him, each, the, themselves, until, below,
are, we, these, your, his, through, don, nor, me,
were, her, more, himself, this, down, should, our,
```

**Figure 14** List of stopwords.

Using NLTK library to deal with stopwords. NLTK contains stopwords list for many other languages than English. Following code snippet shows usage of stopword library.

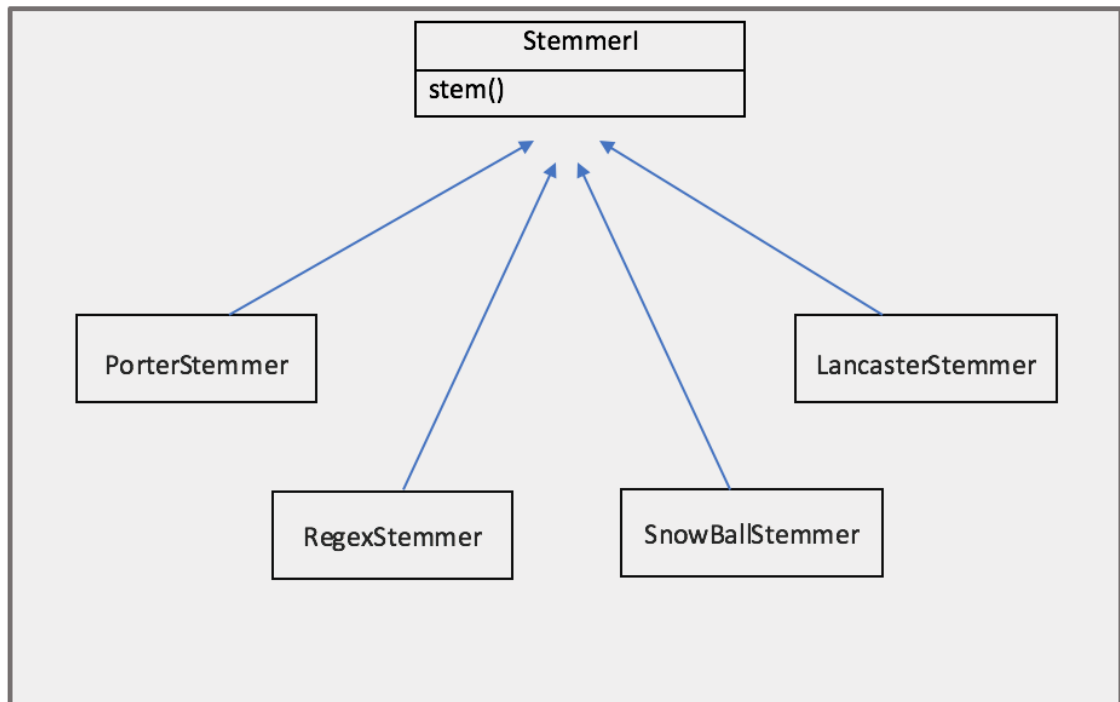
```
>>> from nltk.corpus import stopwords as sword
//loading stopword library
>>> english_stopwords = set(sword.words('english'))
>>> words_list = ["Don't", 'this', 'is', 'available']
//dropping stopwords from string
>>> [ww for w in words_list if w not in english_stopwords]

["Don't", 'available']
```

#### 4.1.5 Lemmatization

*Stemming*, it is the process of normalizing the word into base or single form. Usually, it includes the removal and identification of suffixes, prefixes. For e.g. a simple stemming algorithm will normalize *played, playing, plays* into *play*. On the other hand, *lemmatization* is an advanced form of stemming approach. Lemmatization group words by relying on the core-concept of word. It uses grammatical information to determine the core-concept of word and also uses context surroundings. So, it requires more information to work accurately. If we take a word like *walk* then stemming and lemmatization will

produce same result for that but if we take ‘*meeting*’ which can be used either as verb or noun then stemming will produce *meet* while lemmatization will produce *meet* for verb and *meeting* in-case of noun. Figure 15 shows list of algorithms which can be used with *StemmerI* interface. PorterStemmer and LancasterStemmer are widely used in lemmatization. Major difference between these two algorithms is that Lancaster stemming involves the use of more words of different sentiment as compare to Porter stemmer [44].



**Figure 15** *Inheritance of StemmerI interface*

Following code snippet shows usage WordNetLemmatizer class to produce lemma. Its clearly visible that lemmatizer changes the result if part of speech value is added.

```

>>> from nltk.stem import WordNetLemmatizer
//loading lemmatizer
>>> lemma = WordNetLemmatizer ()
//results on single word using different approach
>>> lemma.lemmatize('booking') → 'booking'
>>> lemma.lemmatize('booking', pos='v') → 'book'
>>> lemma.lemmatize('books') → 'book'
  
```

#### 4.1.6 Spelling Correction

In natural language processing task, we have huge amount of text data which is mostly raw data collected from social media site, blogs etc. Misspelled words can result in

increasing the size of vector space. When dealing with large corpus this issue can take more resources and time [45]. There are multiple approaches to correct the spellings automatically. For e.g. dictionary based approach, fuzzy matching algorithms like *metaphone*, *soundex*. Those string hashing methods like metaphone or soundex are already being used in text mining tasks. Also, these approaches can be used together. In spelling correction task, *PyEnchant*<sup>13</sup> library is being used, it's a spellchecking library for Python. While processing PyEnchant checks whether the word is present in the dictionary or not. Following piece of code shows the usage of PyEnchant library.

```
>>> from replacers import SpellingReplacer
// instantiating spell replacer library
>>> Spellreplace = SpellingReplacer()
//correcting the word if it's in library
>>> Spellreplace.replace('cookbok') → 'cookbook'
```

## 4.2 Sequential Data

As mentioned earlier that we will be using Long Short-Term Memory for this task. LSTM has the ability to learn and remember the long sequences of inputs. If non-sequential data is taken as input then it ignores any spatial correlation between words and the placement of words according to each other. So, this text representation doesn't retain order of words. We need a desirable and more advanced representation method. Current best approach is to present text in sequence form, this way we can retain order of words. We can retain order by creating a vector of equal length or smaller. To get vector of same length for each sentence, max length should be defined. Making vectors of same length can be achieved by cutting the vectors (removing redundant parts) and pad the vector with 0's which are short in length.

By using the presentation shown in figure 16, each word can be represented as a vector. To get sequential data and to meet other requirements we have used Keras Tokenizer<sup>14</sup>, it has other built-in functionalities like:

- `fit_on_texts(texts)` → for tokenizer training
- `texts_to_sequences(texts)` → to convert text into sequence

For padding the short sentence, we have used Keras Sequence Preprocessing<sup>15</sup>. `sequence.pad_sequence(sentence, maxLen)` → to generate padded sequence

<sup>13</sup> <http://pythonhosted.org/pyenchant/>

<sup>14</sup> <https://keras.io/preprocessing/text/>

<sup>15</sup> <https://keras.io/preprocessing/sequence/>

### An example of word sequence:

Let assume our text corpus contains two sentences:

1. Emma performed well in Noah
2. Emma deserves an award.

If we use dictionary approach then:

- emma : 1
- performed : 2
- well : 3
- in : 4
- noah : 5
- deserves : 6
- an : 7
- award : 8

Sequences of words for these sentences will be if we choose our max vector length 5:

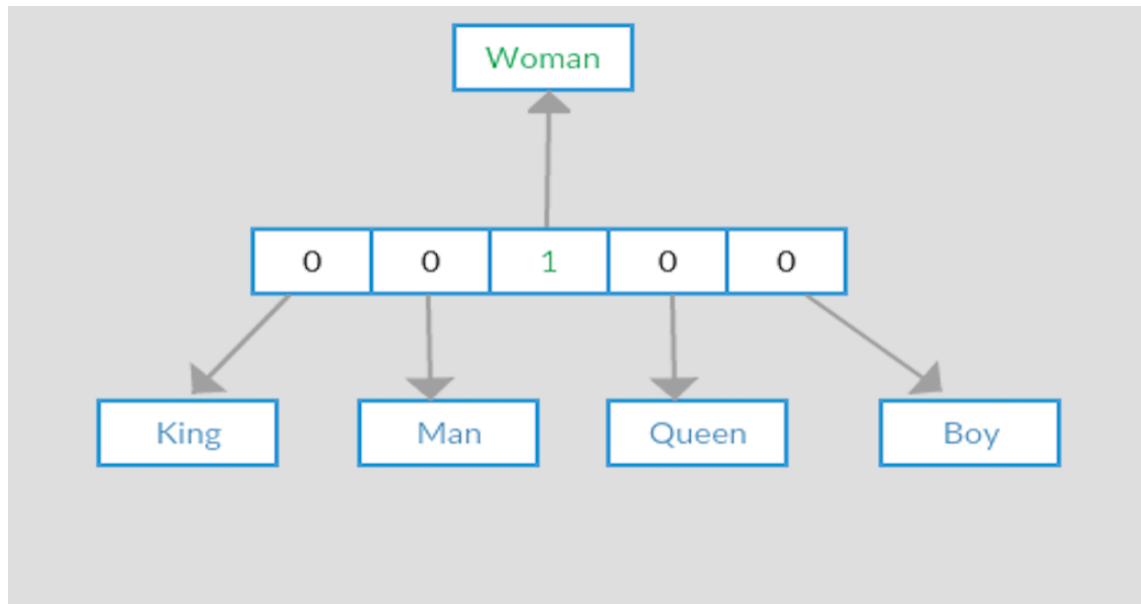
1. [1,2,3,4,5]
2. [1,6,7,8,0]

In second sequence, we have padded short sentence with '0'.

*Figure 16 Word vector approach example*

## 4.3 Word Vectors

Word vector is a vector of weights. Inside vector every element is associated to word in 1-to-N encoding. In word encoding, element which corresponds to word is set to 1 and rest of them to 0. Suppose, there are five words in our vocabulary. King, Man, Woman, Queen, Boy. So, encoding for word Woman shown in figure 17.



**Figure 17** 1-to-N encoding

By using one-hot encoding there is no option left for vector comparison except performing equality testing. But by presenting each word as a vector will incorporate more complex information into representation. This way we will get more information from their placement relative to each other and interaction between them. We can either train our own word representation for each word or some pre-trained models can be used, Google word2vec<sup>16</sup> and GloVe<sup>17</sup>. GloVe, we can get vector representation of words from it and it's an unsupervised learning algorithm [46] where word2vec is another option to do the same thing but the difference is how they are built. GloVe is a 'count-based' model whereas word2vec is a 'predictive' model. word2vec had significant improvements to NLP. Still, there is a room for improvement in word2vec [47]. Word vectors captures semantic and syntactic linguistic regularities [47]. So, word vectors are useful to get features for problems related NLP.

Following example shows that vectors can get semantic similarities among words:

Suppose {"man", "queen", "woman", "king"} are the words represented in vector.

$v_{man}, v_{king}, v_{woman}, v_{queen}$

Relationship will be:

$$v_{king} - v_{man} + v_{woman} \approx v_{queen}$$

Another example:

$$v_{Helsinki} - v_{Finland} + v_{France} \approx v_{Paris}$$

*Note: Shown results were reported by Mikolov [47].*

<sup>16</sup> <https://code.google.com/archive/p/word2vec/>

<sup>17</sup> <https://nlp.stanford.edu/projects/glove/>



These are pre-defined models but to get better results than pre-trained models needs to be trained with domain specific data. Of course, to outperform the pre-trained vectors and to get the domain specific word vectors we need a lot of training data and training time.

### 4.3.1 Keras Embedding Layer

Embedding<sup>18</sup> layer is used on text in neural networks. Embedding layer converts positive integers into vectors. This requires integer encoded data as input so that each word has a unique integer. Keras Tokenizer API can be used to perform this step of data preparation. An embedding layer is initialized with random weights and then it will learn embeddings for all words in training dataset. This layer has always to be the first layer while model preparation and it requires three mandatory arguments.

- **Input\_dim**: it's the size of vocabulary in text data. Suppose our data is integer encoded in range of 0-10 then 11 would be the size of vocabulary. In our case we are calculating this value at runtime.
- **Output\_dim**: it's the size of vector which we will get as output for each word from this layer. This value could be 100, 64, 32 or even larger so try this with different values for defined problem. In our case we have `EMBEDDING_SIZE = 128`
- **Input\_length**: it's the length of sequences as input. In our problem, its `MAX_SENTENCE_LENGTH = 558`

### 4.3.2 Word2Vec Embedding

As defined earlier it's an algorithm to learn word embeddings from a text data. Training algorithms for this approach are *skip gram* and *continuous bag of words*. We didn't get into too much detail of this algorithm as we don't have enough data to train and get domain specific word embeddings but the way it works is, it looks at a window of words for each target word to provide context. We have tested this word2vec in our problem by using Gensim<sup>19</sup>. Gensim provides a word2vec class to use word2vec model. It just involves organizing and loading text into sentences and passing those sentences to the constructor of word2vec() instance. Make sure that sentence has been tokenized. This constructor has many parameters but there are few which we can configure according to our requirements.

- **Size**: default value is 100, it's the dimension of embedding's
- **Sg**: training algorithm to use *CBOW* (0) or *skip-gram* (1) and default value is 0
- **Min-count**: its minimum number of words to consider while training model also has default value 5
- **Window**: it's the maximum distance words around the target word and target word. Default value is 5
- **Workers**: it's the number of threads while training. Default value is 5

But default values are enough but if one has systems with huge hardware capabilities then go ahead and configure those arguments accordingly.

<sup>18</sup> <https://keras.io/layers/embeddings/>

<sup>19</sup> <https://radimrehurek.com/gensim/>

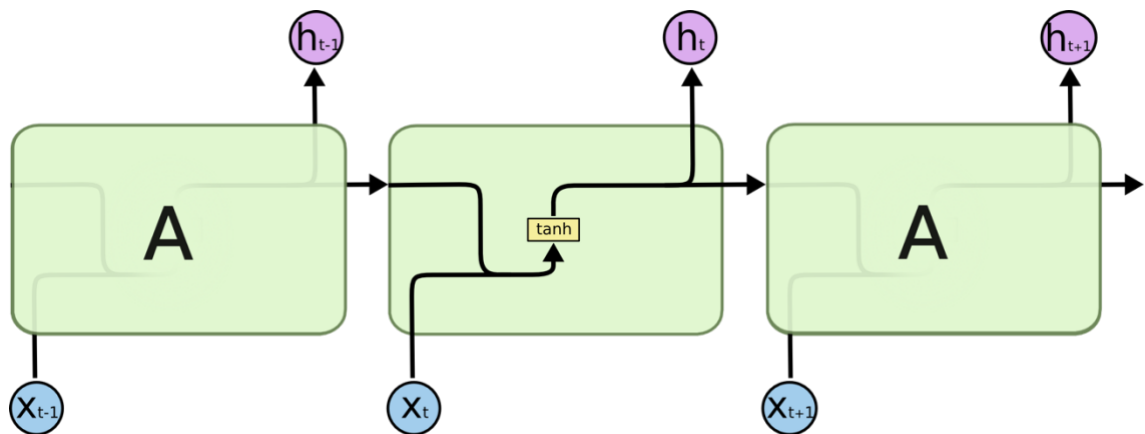
## 4.4 Model Architecture

There are multiple ways to build Neural Network, by using multiple layers and different components. Following are the components and layers used in building the model for this sentiment analysis task.

### 4.4.1 LSTM Layer

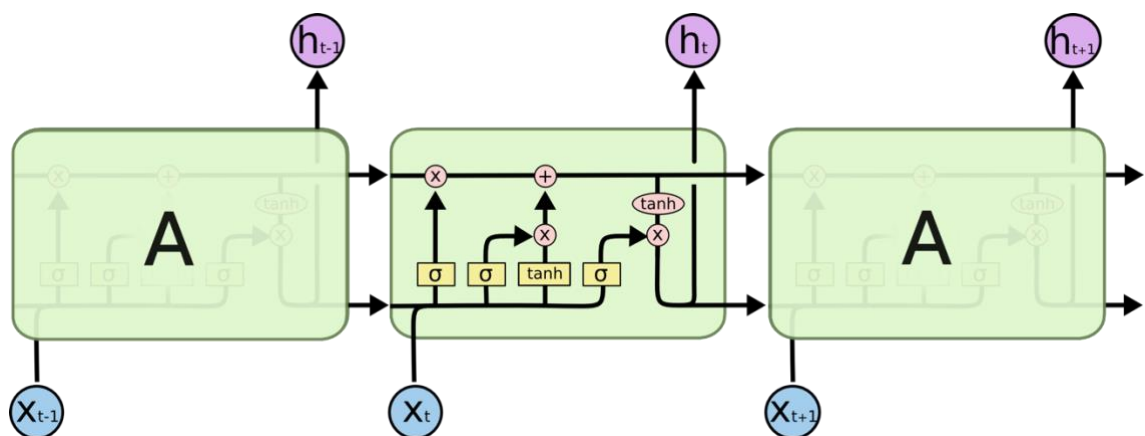
*Long Short-Term Memory* network is a special kind of recurrent neural network was proposed by Hochreiter & Schmidhuber [6] as an extension to recurrent neural network. LSTM networks are capable of learning long-term dependencies. It's the default behaviour of LSTM network to remember information for a long period.

A recurrent neural network has a chain of repeating modules of neural network. Figure 18 [6] shows the structure of typical RNN. Each block in the figure is called a module.



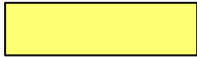




*Figure 18 Repeating modules in Recurrent Neural Network.*

LSTM also has this same chain structure but blocks have different structure than RNN. There are four neural network layers instead of single in a module. Those layers are interconnected in a special way.



*Figure 19 LSTM module contains four interacting layers.*

Following notations will be used to understand the *Figure 18* [6].

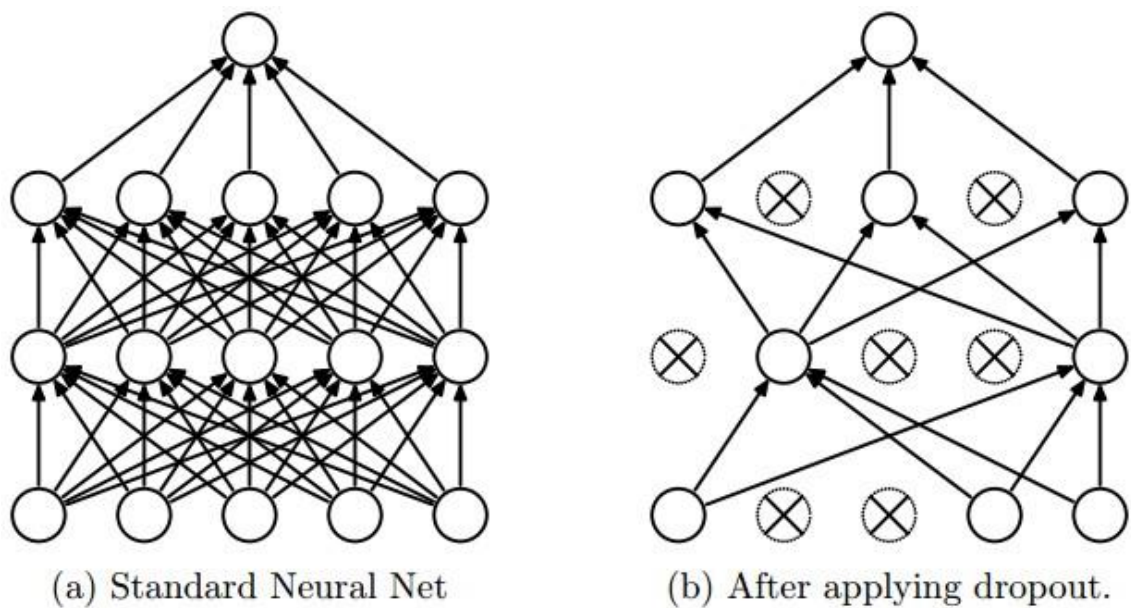
- Neural Network Layer 
- Pointwise Operation 
- Vector Transfer 
- Concatenate 
- Copy 

In Figure 19 each line holds the word vector, usually from the output of block to the next one as input. Pointwise operations like vector multiplication or addition are performed by pink circles. Yellow rectangles are layer of learned neural networks. Lines merging with each other are performing concatenation. Contents are being copied and going to different locations with forking line where  $X_t$  is the input at time step  $t$  and  $h_t$  is the output at time  $t$ .

#### 4.4.2 Dropout Layer

Dropout technique used to avoid over-fitting [48]. Learning the training data too well then, it's called over-fitting. While training when a model learns the noise and detail to an extent that it impacts the performance negatively on new data.

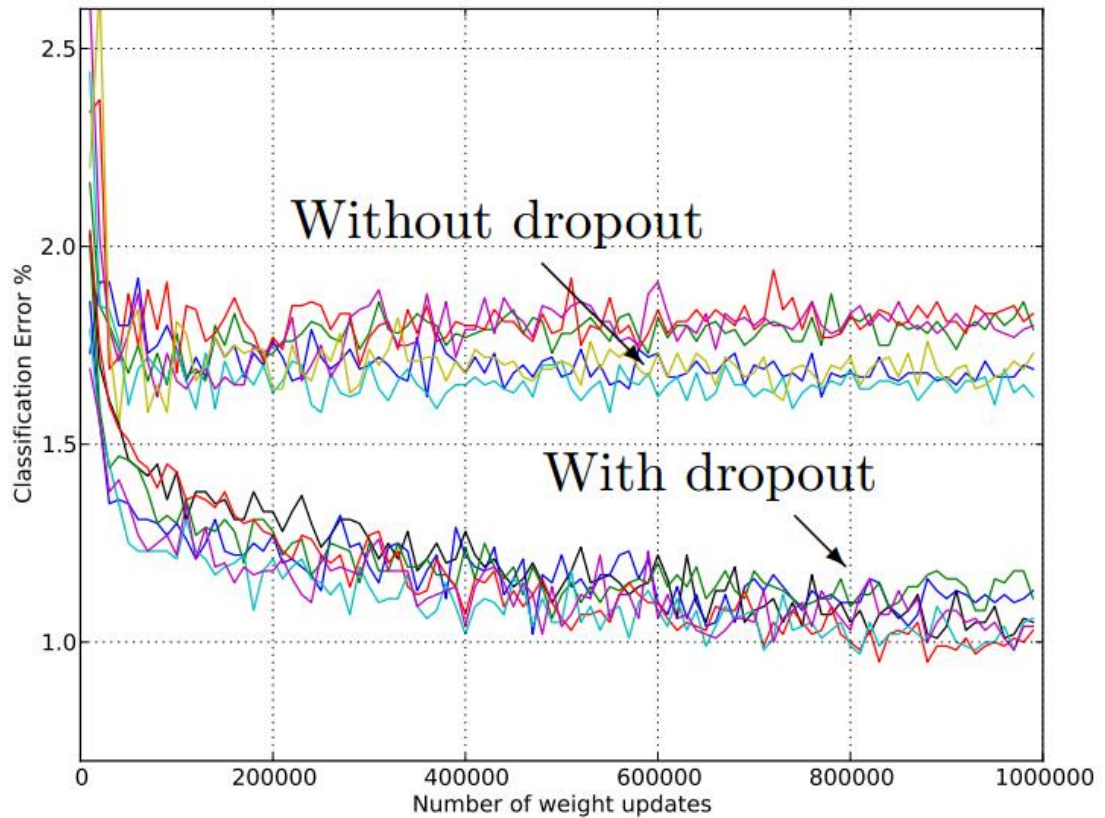
During training on a specific layer half of the neurons switched off. This process improves generalization because it forces the layer to learn the same "concept" with different neurons.



*Figure 20 Network before and after Dropout*

Figure 20.a is the network before the dropout process. All neurons are connected to other neurons while Figure 20.b shows the deactivated neurons. Dropout is deactivated in the prediction phase.

Dropout helps model to increase accuracy and generalize better [48]. Figure 21 illustrates the effect of with and without dropout.



**Figure 21** *Dropout effect on training of network*

As we can see that classification error (at Y-axis) is decreasing with dropout which proves that model training is improving with iterations.

#### 4.4.3 SpatialDropout1D

Functionality of `spatialdropout1D` is same as dropout but it drops the whole one-dimensional *feature map* rather than dropping elements individually.

- Feature map is a process of mapping input features to hidden units to create new features. Those new features are then fed to the next layer.

When frames inside feature maps are strongly connected to each other, then regular dropout doesn't work. In this case `spatialdropout1D` works better.

#### 4.4.4 Batch Normalization

Batch normalization is a technique to improve overall accuracy of neural network and it also increases the learning performance [49]. Idea of this technique is to normalise input of layer by shifting standard deviation to '1' and mean output activation to '0'. Neural

network is a combination of multiple layers. Output of one layer becomes an input of next layer. Normalization is performed on the output of a layer before applying activation function on it and then feed that processed output to next layer.

#### Advantages of Batch Normalization:

- **Fast Network Training:**  
Because of extra normalization calculation each iteration for training will be slower but the overall training will be faster.
- **Easier Weight Initialization:**  
Initialization of weight is a difficult process especially creating deep networks. Batch normalization reduces the initial starting weigh sensitivity.
- **Adds Regularization:**  
Normalization adds little noise to the network which increases the learning capability. In some cases, batch normalization performed as well as dropout [50].

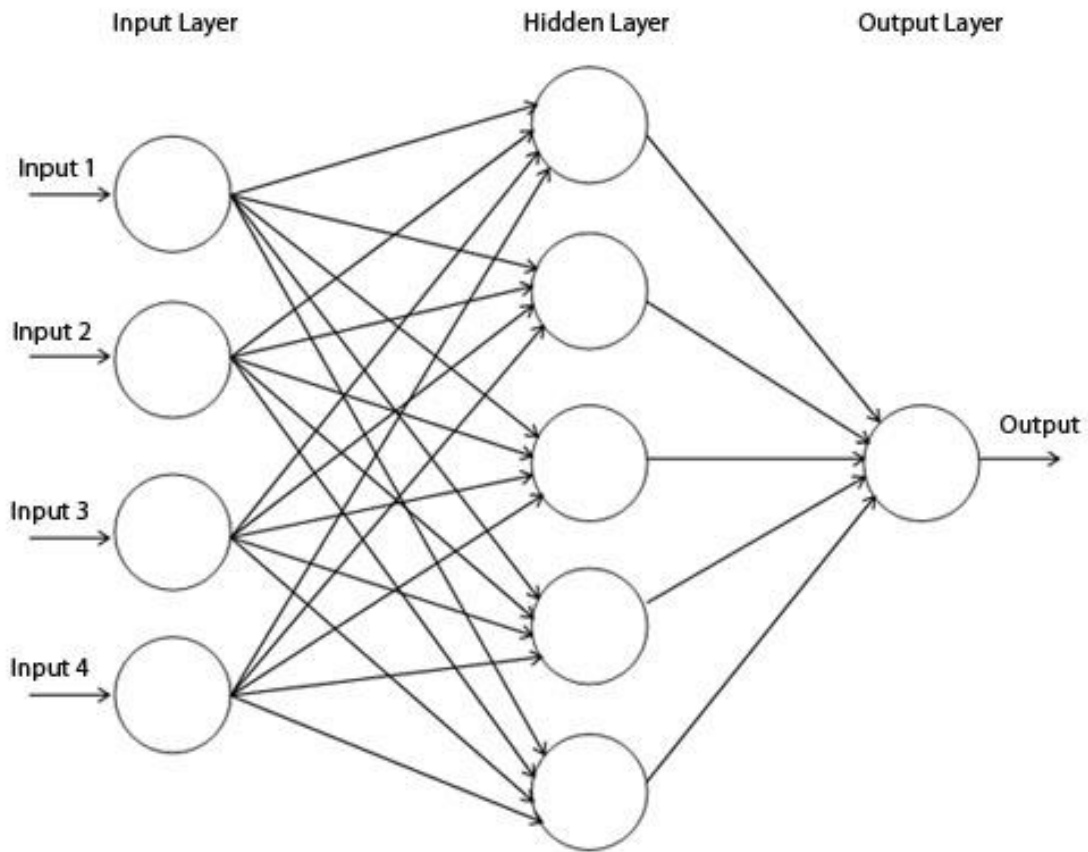


*Figure 22 Performance with and without batch normalization*

Figure 22 illustrates that batch normalization really helps in network training but not always [51]. Blue line Green and Red line have higher accuracy value (Y-axis) than Blue one.

#### 4.4.5 Dense Layer

Dense layer is a simple layer in which each neuron is connected to each neuron in next layer. Usually it is followed by a no linear activation function.



**Figure 23** Example of Dense layer in Feedforward neural network

Figure 23 is the architecture of *feedforward network* (term *feedforward* means no backward connections allowed). In feedforward network connections between nodes is only allowed from nodes in  $i$ -layer to  $i+1$ -layer. All arrows pointing in one direction and each neuron is densely connected to neuron in next layer. A dense layer is a fully connected layer [52].

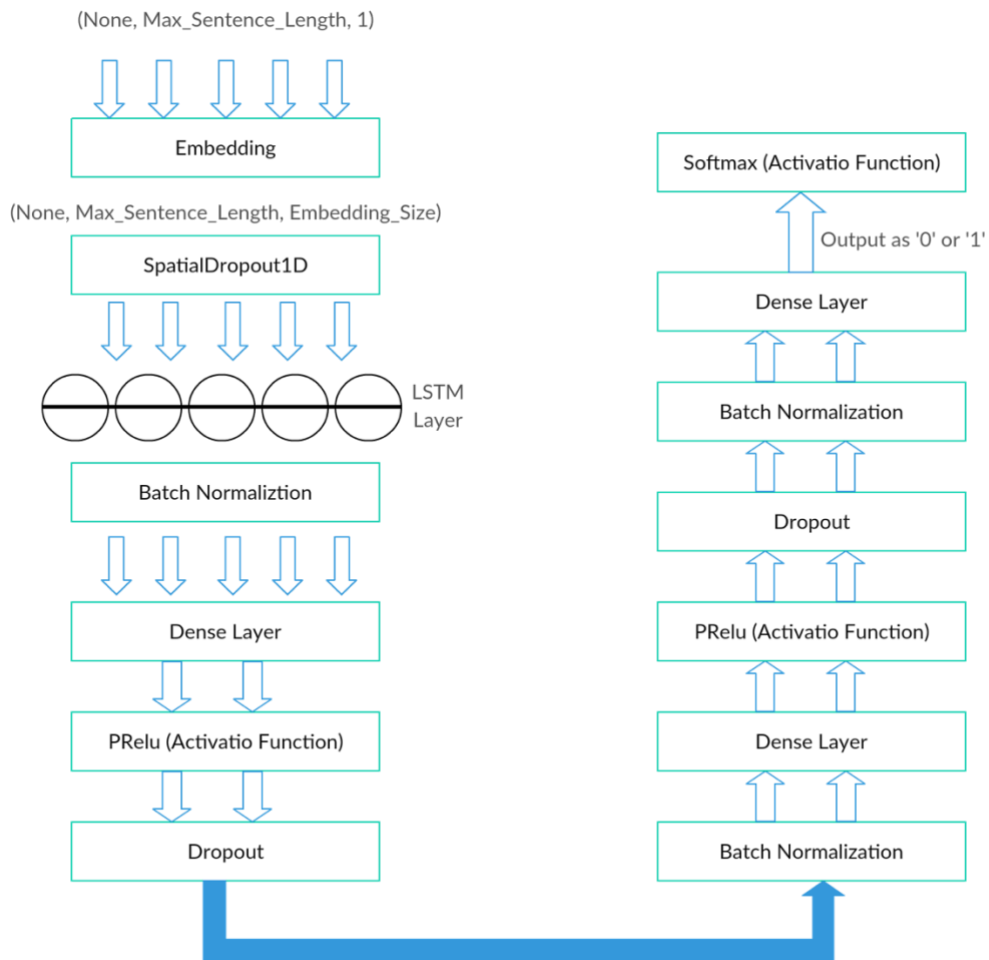
#### 4.4.6 Network Topology

Above mentioned layers and components has been used to build the model for sentiment analysis task. Figure 24 shows the network topology of model. In this model input for each row is a sequence of word indices and the length of sequence given by `MAX_SENTENCE_LENGTH`. First dimension of the *tensor*<sup>20</sup> is set to None to mention that batch-size<sup>21</sup> is unknown at definition level. Batch size value will be specified at runtime using `BATCH_SIZE` parameter. With undetermined value of batch size, shape of input tensor would be:

(None, `MAX_SENTENCE_LENGTH`, 1)

<sup>20</sup> Data object or array

<sup>21</sup> Number of records fed to network each time



**Figure 24 Model network topology**

These tensors are fed to Embedding layer of size EMBEDDING\_SIZE. Embedding size is calculated according to requirement and complexity of task. Weights of embedding layer are learned during training time but in start they are initialized with random small values. Embedding layer will transform tensor to a new shape:

$(None, \text{MAX\_SENTENCE\_LENGTH}, \text{EMBEDDING\_SIZE})$

Embedding layer output will be fed to LSTM layer but before feeding to LSTM the tensor will be passed through SpatialDropout1D (As explained in section 4.4.3). Tensor shape from LSTM output would be:

$(None, \text{HIDDEN\_LAYER\_SIZE}, \text{MAX\_SENTENCE\_LENGTH})$

Further, tensors from LSTM layer processed with Batch Normalization, Activation functions and fed to multiple Dense layers. Final Dense layer has only one neuron so output size would also be one, so either, it will be '1' positive review or '0' negative review.

Two Dense layers have been used in the model but there is no rule of thumb to find out efficient and reasonable network topology by calculating the quantity of inputs and outputs. It depends upon the complexity of classification and number of training examples. Add the layers until test error is not improving [53]. In the process of learning features, input layer learns edge detectors and rest of the layers learns complex features [54].



## 5. EXPERIMENTATION AND RESULTS

This section will expose the evaluation metrics to measure the performance of the model and scale at which *hyper-parameter*<sup>22</sup> tuned.

### 5.1 Hyper-parameter Optimization

The computational and classification performance strongly depends upon number of hyper-parameters used and their tuning level, some of hyper-parameters are interdependent to each other. Tuning hyper-parameters is a time-consuming process as they require multiple experiments and even after tuning, results can be inconclusive. There could be different reasons for such unexpected results. For e.g. not strictly deterministic implementation of some routines or random correlations. Some of the important hyper-parameters discussed in following paragraphs:

#### 5.1.1 Embedding dimension:

Word-embedding dimension helps to determine how much different aspects of syntactic and semantic information each word vector can represent. In this sentiment analysis work, not a reasonable result was experienced by using EMBEDDING\_SIZE value more than 64. So, in all experiments EMBEDDING\_SIZE value fixed to 64.

#### 5.1.2 Number of Neuron:

Learning capacity of network depends upon number of neuron being used. Using excessive number of neurons result in learning more structure of problem at cost of long training time and also can lead to over-fitting. Similarly using less number of neurons will lead to a term called *under-fitting*<sup>23</sup>.

There are a lot of rule-of-thumb methods to calculate number of neuron to be used in hidden layers, such as following but still those rules are not authentic as quantity of neuron varies from problem to problem:

- Number of neurons should be between the size of input and output layer.
- Number of neurons should be between two-thirds (2/3) of input layer.
- $n < INPUT\_LAYER\_SIZE * 2$   
 $n$  = number of neurons  
 $INPUT\_LAYER\_SIZE$  = size of input layer  
 Neuron should be less than double the size of input layer.

<sup>22</sup> A configuration which is external to model and value of hyper-parameter cannot estimated from data.

<sup>23</sup> Model that cannot model the training data properly also unable to generalize to new data.

For this task, equation 5.1 [55] helped in start for getting reasonable neuron but number of neurons changed during different experiments.

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))} \quad (5.1)$$

$N_i$  = no. of input neurons

$N_s$  = no. of samples in training data

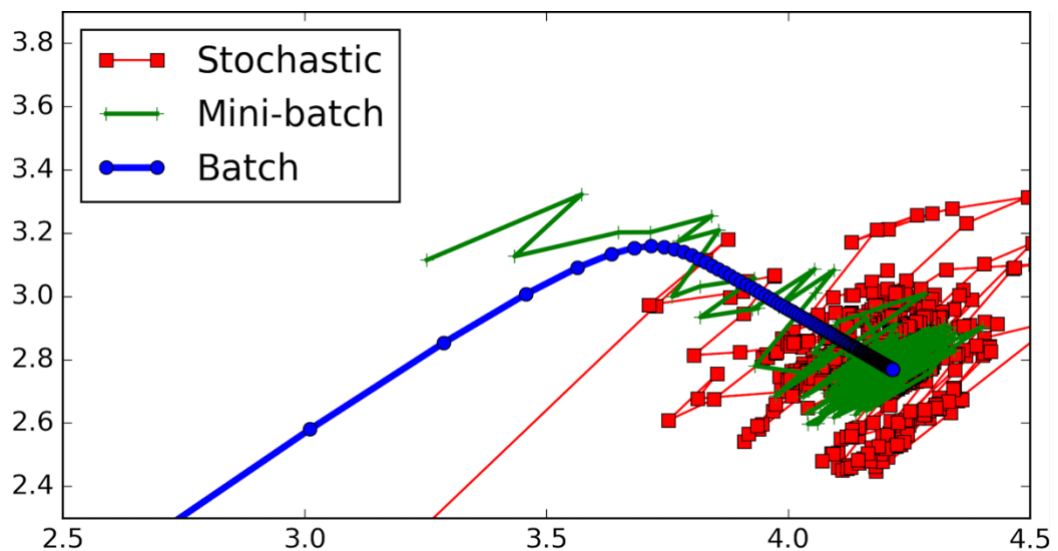
$N_o$  = no. of output neurons

$a$  = arbitrary scaling factor between 2-10.

### 5.1.3 Batch Size:

Batch size handles how often to update the weights in network. Over an entire batch of inputs *objective function*<sup>24</sup>  $f$  is an average so, using large value of batch-size reduces variability of parameter values over time. Large batch-size affects performance of computation, as weights are not being changed frequently.

It defines quantity of input samples that will be propagated through the network at once. For e.g. dataset contains 550 training samples and *BATCH\_SIZE* value is 100. Algorithm will take first 100 input samples from dataset and trains the network. In next turn algorithm will take next 100 input samples and train the network again. Algorithm will repeat the process until it processes entire dataset.



**Figure 25** Gradient descent calculation with different batch-size value

There are some pros and cons of using small batch size [56].

#### Pros:

- Small batch-size uses less memory. Training a network using less number of samples is helpful when it's difficult to fit dataset in memory.
- Using mini-batch helps network to train quickly.

<sup>24</sup> A general term for any function that we optimize during training.

### Cons:

- Using smaller batch-size decreases the accuracy of estimating the *gradient descent*<sup>25</sup> value.

Figure 25 clearly shows that mini-batch (small batch-size) direction is fluctuating for gradient's estimation. Gradient is changing its direction even more in stochastic as batch-size for stochastic is  $l$ .

After testing multiple sizes between 20 ~ 64, batch-size for our model fixed to 32.

### 5.1.4 Dropout-Rate:

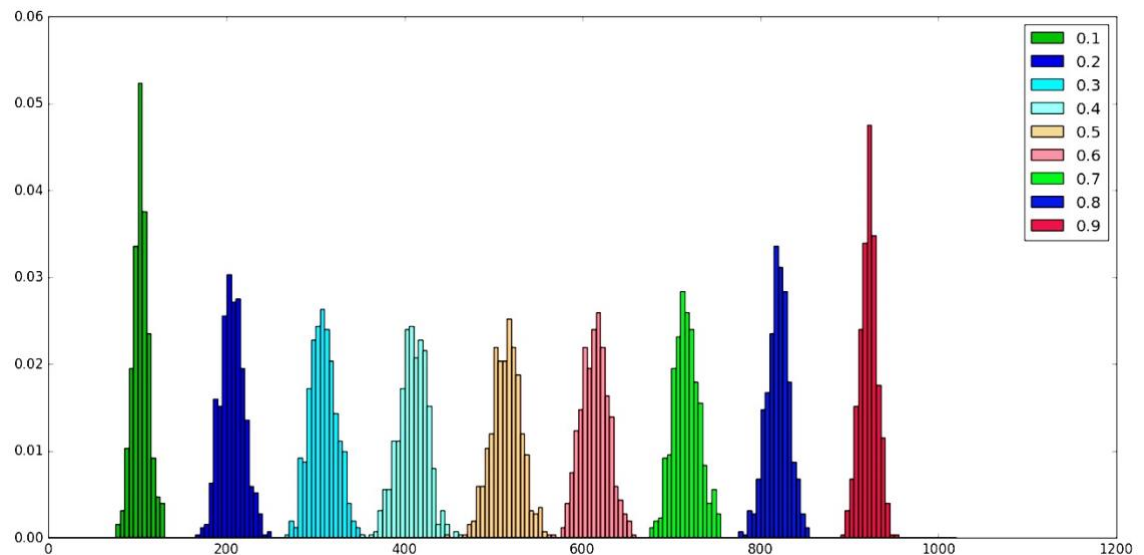
Setting dropout-rate ' $r$ ' greater than zero results in increasing the number of training epochs to reach *local maxima*<sup>26</sup>.

While using Dropout (explained in section 4.4.2), dropout-rate fixed to a value  $p$  for a layer which results in dropping proportional number of neurons from layer. For example, Dropout layer has neurons  $n = 1024$  and  $p = 0.5$ . It means that 512 neurons will get dropped. Let's verify this statement using following binomial equation [57]:

$$Y = \sum_{i=1}^{1024} X_i \sim Bi(1024, 0.5)$$

$$P(Y = 512) = \binom{1024}{512} 0.5^{512} (1 - 0.5)^{1024-512} \approx 0.025$$

so, the probability of dropping 512 neurons is only 0.025.



**Figure 26** Probability ratio of dropping out neurons

Figure 26 shows the probability of dropping neurons with fix number of neuron  $n=1024$  and different values of  $p$  ( $l \sim 10$ ). Number of neurons at X-axis and probability of

<sup>25</sup> Optimization algorithm used for calculating the coefficients or weights.

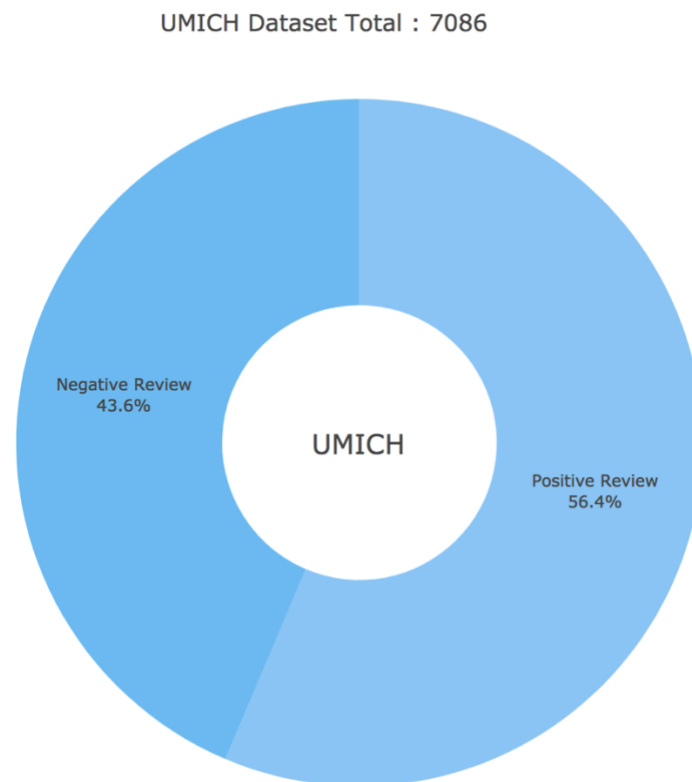
<sup>26</sup> Greatest value in a set of points but it's not a highest in-comparison with all values in set

dropping neurons at Y-axis. Still dropout-rate also depends upon the complexity of problem. In some cases, increasing dropping probability helps to achieve good results. Typically, it's value is used 0.2 but in this task dropout-rate fixed to 0.3 and 0.5 as two Dropout layers has been used.

## 5.2 Results:

This section presents the results obtained from sentiment analysis. Experiments were conducted for the three kinds of datasets. An overview of results can be found in Table 4.

### 5.2.1 Performance on UMICH Dataset:



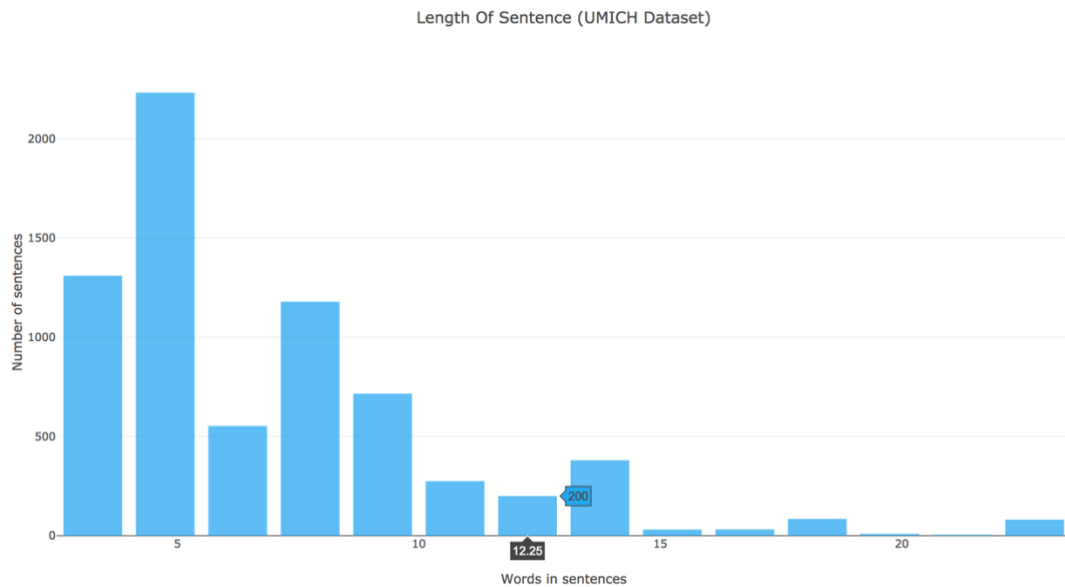
**Figure 27** *No. of positive and negative reviews in UMICH dataset*

Figure 27 shows classification of UMICH dataset which contains total 7086 number of samples from which 3089 are negative and 3997 are positive. This dataset was used in a Kaggle<sup>27</sup> competition and already labelled.

Further figure 28 tells us other details of dataset, for e.g. the number of words in a sentence (length) and quantity of sentences containing with different length. This helps us to measure the capability of LSTM to remember history. The information visualised

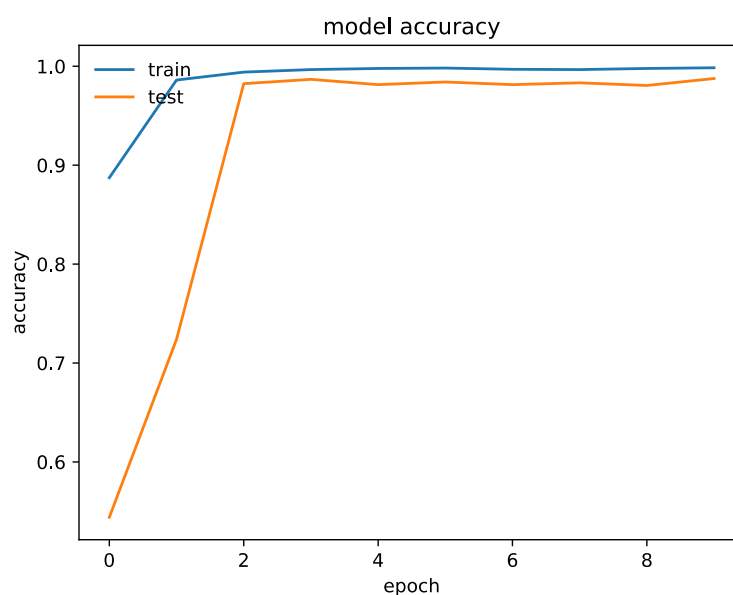
<sup>27</sup> <https://www.kaggle.com/>

in Figure 27 and 28 is before pre-processing the dataset. Bars for long sentences are not high which mean dataset doesn't contain too much long sentences.

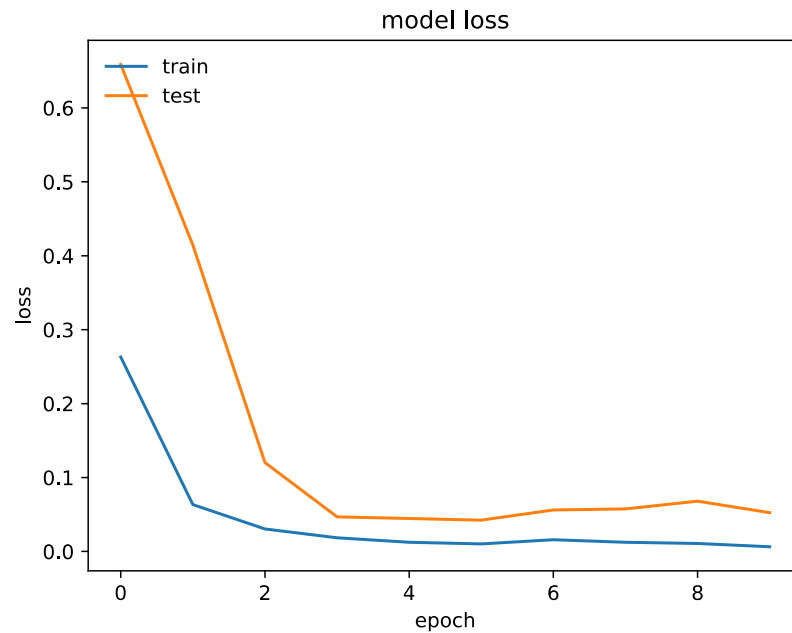


**Figure 28** *Insight of UMICH dataset*

Performance of model trained with UMICH dataset is visualized in Figure 29. Class distribution among the dataset was quite reasonable and model performed quite well and achieved 98% accuracy. Model was validated during training by a validation set which was created by splitting the data with a ratio of 80: 20 (training-set: validation-set) further model was tested by totally unknown data to model which was taken from training dataset before training. This technique was used with all our experiments.



**Figure 29** *Model accuracy on UMICH dataset*



**Figure 30** Error (loss) rate on UMICH dataset

From Figure 30 it's clearly visible that accuracy increased for training data after first epoch (iteration) but it's still increasing on test data after first epoch. After second epoch, there is no significant change in both lines as it's already achieved 98% accuracy. Similarly, in Figure 30 both lines are dropping down towards 0 which is an indication of good learning after 3<sup>rd</sup> epoch loss rate <sup>28</sup>increased a little bit but that doesn't affect the overall performance. From UMICH results, mentioned in table 1 we conclude that model structure this dataset is too much big that model learned the features so quickly. Table No.1 shows the prediction results:

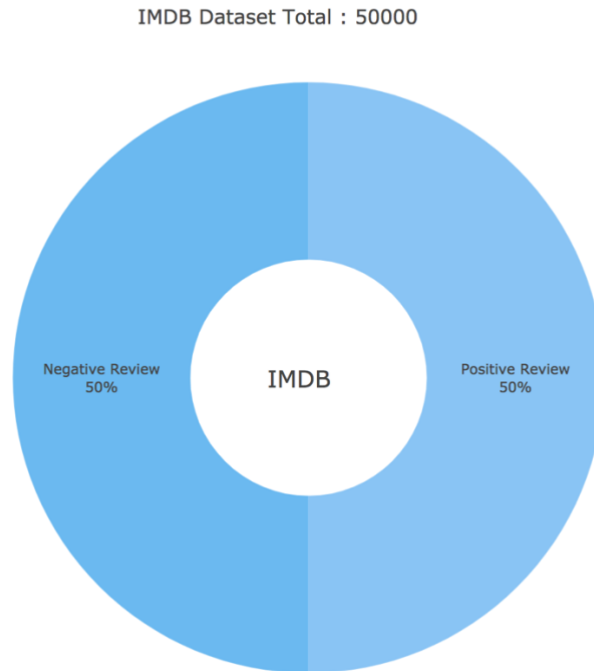
**Table 1.** Model prediction on UMICH dataset

LABELLED VALUES	PREDICTED VALUES
0	0
0	0
1	1
0	0
1	1

### 5.2.2 Performance on IMDB Dataset:

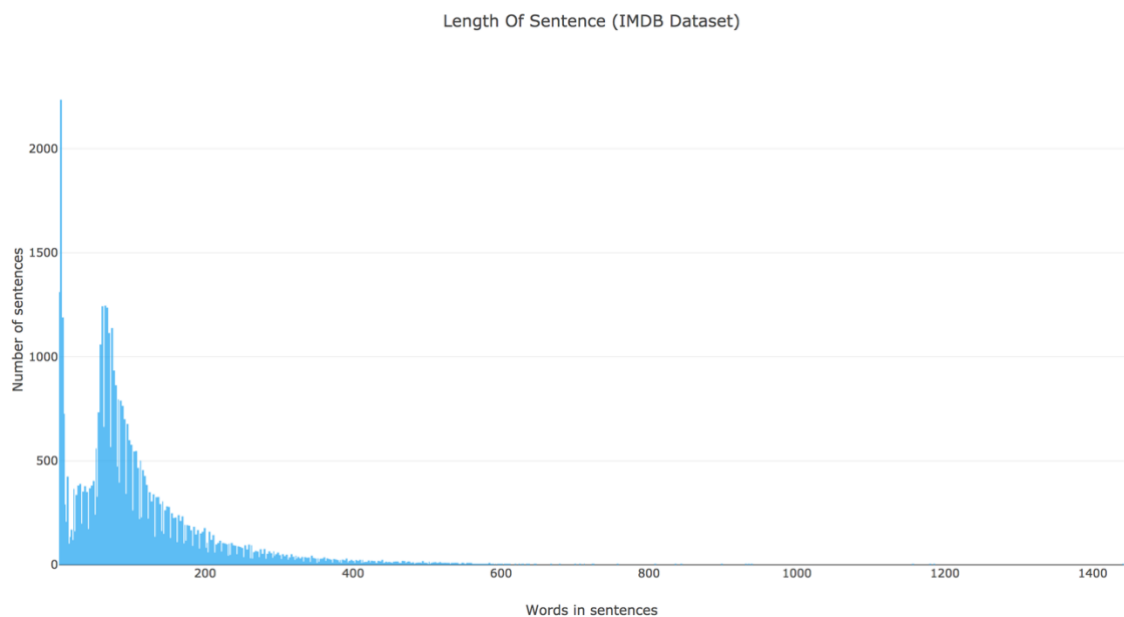
As IMDB datasets exhibit very balanced distributions among the positive and negative reviews in Figure 31. Total number of samples are 50,000 from which 25,000 were positive and negative. This dataset was labelled and modified by Stanford university [9].

<sup>28</sup> Loss is calculated on validation and training and it tells, how well the model is performing.

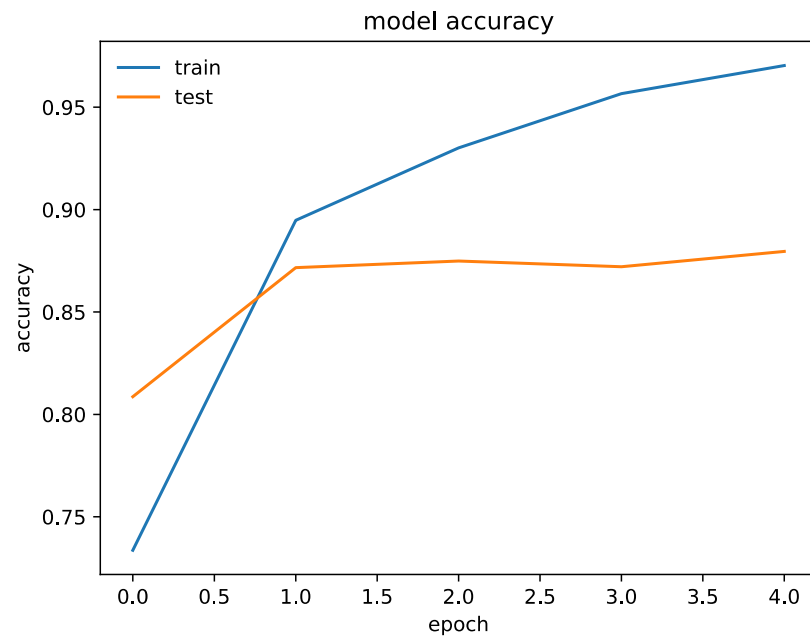


**Figure 31** *No. of positive and negative reviews in IMDB dataset*

In Figure 32 the bars for very short sentences are high which means dataset contains a lot of small sentences. But the frequency of medium sized sentences is very reasonable as in our case a paragraph containing 20 to 200 words is good which can help to analyse the history remembering capability of model. In this data, we have also very huge paragraphs containing max 1400 words but they are less in quantity.



**Figure 32** *Insight of IMDB dataset*



**Figure 33** *Model accuracy on IMDB dataset*

Performance of model trained with IMDB data visualised in Figure 33. Model learning on training data is increasing gradually after first epoch but according to test data our model is learning much because its good only in training data but not showing good results on unseen data. But still model achieved an accuracy of 96% with this dataset.



**Figure 34** *Error (loss) rate on IMDB dataset*

Figure 34 illustrates the loss rate of model on IMDB data and it's clearly visible that our model is over-fitting on this dataset. As loss rate is increasing heavily after first epoch. It



means model isn't performing well on test data but learning on training is very good. Simple understanding for over-fitting and under-fitting is:

$$\textit{training loss} \ll \textit{validation loss} \quad \textit{Over-fitting}$$

Similarly:

$$\textit{training loss} = \textit{validation loss} \quad \textit{Under-fitting}$$

Table 2 shows the actual labelled values and predicted values.

**Table 2. Model prediction on IMDB dataset**

LABELLED VALUES	PREDICTED VALUES
1	1
1	0
1	1
0	0
0	0

### 5.2.3 Performance on Amazon Dataset:

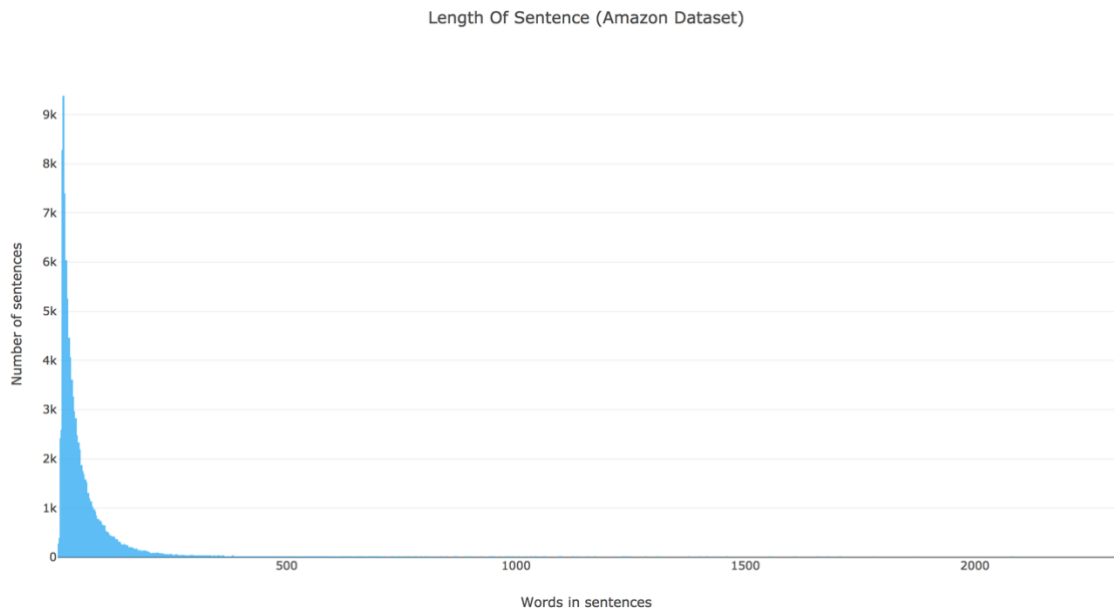
Amazon dataset is publically available, for this task only Product reviews data has been used.

Amazon Dataset Total : 149044



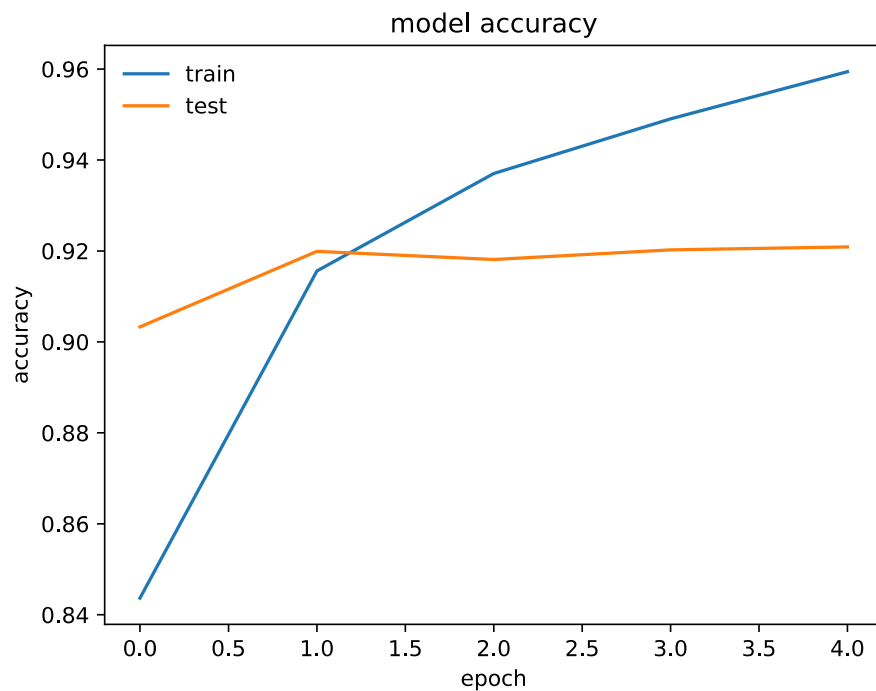
**Figure 35** *No. of positive and negative reviews in Amazon dataset*

This dataset contains 1 lac and around 50 thousand samples. Almost equally divided in both classes (negative and positive) as shown in Figure 35.



**Figure 36** *Distribution of amazon data based on length*

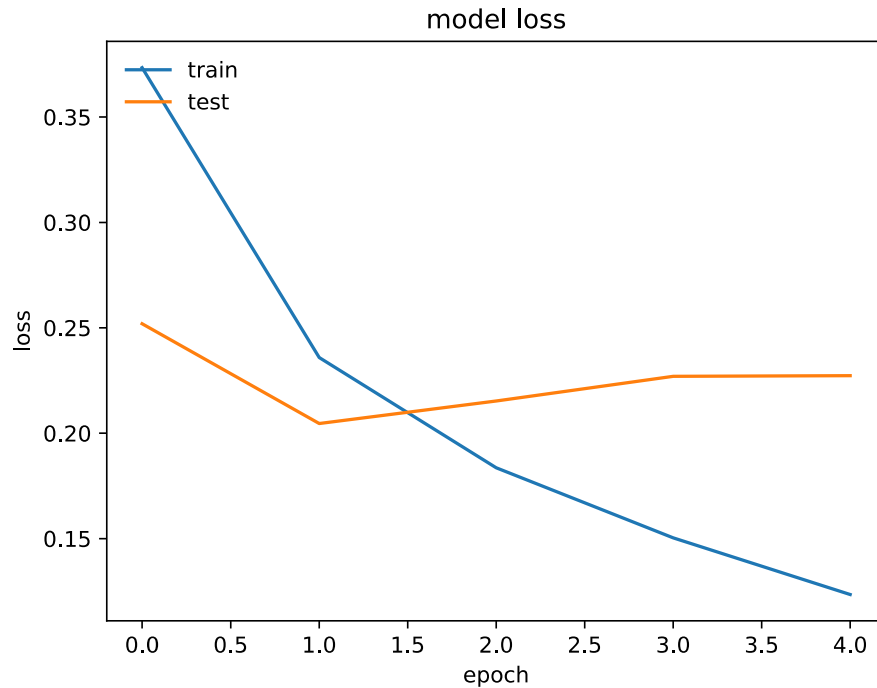
Bars for amazon data lies between 1 to approx. 250 words also frequency of sentences is also hugs in this range (1 ~ 250). Most of the reviews are containing 10 ~ 250 characters in Figure 36. There are some outlier reviews in data with more than 2000 words. Typically, reviewers write maximum half a paragraph.



**Figure 37** *Model accuracy on Amazon dataset*

Even with 92% accuracy on amazon data our model performance is not good. Still, there is over-fitting. Typically gap between validation and training data in Figure 37 indicates

the amount of over-fitting of model and in this case gap is increasing gradually. There could be multiple reason for this problem. One possible case could be that when test accuracy tracks the training accuracy fairly well then model capacity is not high. So, increasing the number of parameter can be helpful in that case.



**Figure 38** Error (loss) rate on Amazon dataset

Improvements will be linear with low learning rates and on the other hand graph will be more exponential with high learning rates. Figure 38 shows that learning rate is not very good but low which is acceptable in this task. Still there is over-fitting.

**Table 3** Predicted values by model trained with amazon dataset

LABELLED VALUES	PREDICTED VALUES
1	0
0	0
1	1
0	0
1	1

#### Performance Table:

Table 4 shows the result of model on all datasets with some change in parameters.

**Table 4** Analyses of models among different dataset.

Models	Accuracy	Dataset	Embedding Size	Max Sentence Length words
Model	98%	UMICH	128	23
Model	96%	IMDB	250	1400
Model	92%	Amazon	250	2000+

## 5.2.4 Resources Consumed

*Table 5 Resources consumed by model on different datasets.*

Model	Nodes	Core per Node	CPU Efficiency	Memory Efficiency
UMICH	1	4	53.20%	7.76% of 15.62 GB (3.91 GB/core)
IMDB	1	4	67.31%	12.76% of 21.76GB (5.44 GB/core)
Amazon	1	8	91.53%	48.96% of 62.50GB (7.81 GB/core)

## 6. CONCLUSIONS

Sentiment analysis is a challenging field with a lot of hurdles as it includes natural language processing. In this thesis capabilities of LSTM have been demonstrated by analysing multiple different type of datasets.

- a. UMICH
- b. IMDB
- c. Amazon

Model performed very well on UMICH dataset as there were small sentences in the samples but our model learned the features very quickly in that case. In all experiments Keras tokenizer and Embedding's were used. Result would be different if Google word2vec or Stanford's Glove model used for embedding. The focus of analysis was to analyse the capability of LSTM to remember the history. In some cases, it performed well but not in other. There could be other reasons for example Dropout ratio, embedding size, number of layers or neuron quantity. Our model architecture was designed to either take entire paragraph or single sentences as input. Change in length of sentences affected the results. Model performed well with medium size (200 words) sentences. Increasing the length resulted in bad results specially over-fitting. Dictionary based feature approach was implemented to test with LSTM and that approach didn't perform. It's also expensive in terms of time and resource consuming. During experiments network architecture size was increased and decreased to test the performance. For amazon dataset, an extra layer was added because of lengthy data samples. However, not all the deeper networks worked well. Three layer architectures performed poorly than single layer architecture. Especially the three-layer architecture is dependent on a large dataset. Variations of Batch size and Dropout value also effected the results. If the batch size is set to higher value then high variance was shown over epochs. Overall deeper networks need huge amount of training data to achieve good results but also need powerful CPU and GPU to perform training task.

Some directions for future work:

- To increase the accuracy. Parameters needs to increase for example POS tagging, Emoticons understanding according to the situation they used. To understand the significance of sentence, context in which sentence used is important.
- Using lexicon approach may affect the results.
- Hyper-parameter tuning will result in improvement of results for sure.
- Hashtag prediction in classification can be helpful as it's common now to use hashtag to present the expressions.

## 7. REFERENCES

- [1] M. Mäntylä, D. Graziotin and M. Kuuttila, “The Evolution of Sentiment Analysis - A Review of Research Topics, Venues, and Top Cited Papers,” Oulu, Finland.
- [2] S. Mukherjee, “Sentiment Analysis, A Literature Survey,” Bombay, 2012.
- [3] B. Liu, “Sentiment Analysis and Subjectivity,” University of Illinois at Chicago, Chicago, 2010.
- [4] K. Dave, S. Lawrence and D. M. Pennock, “Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews,” NEC Laboratories America, Overture Services, Inc, 2003.
- [5] B. Liu, “Sentiment Analysis and Opinion Mining Synthesis Lectures on Human Language Technologies,” University of Illinois at Chicago, Chicago, 2012.
- [6] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” München, Germany, 1997.
- [7] A. Graves, “Supervised Sequence Labelling with Recurrent Neural Networks,” 2012.
- [8] S. Negi and P. Buitelaar, “Towards the Extraction of Customer-to-Customer Suggestions from Reviews,” 2015.
- [9] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng and C. Potts, “Learning Word Vectors for Sentiment Analysis,” Stanford University, Stanford, 2011.
- [10] B. Pang and L. Lee, “Opinion Mining and Sentiment Analysis,” Computer Science Department, Cornell University, Sunnyvale, 2008.
- [11] P. D. Turney, “Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews,” National Research Council of Canada, CA, 2002.
- [12] B. Liu, “Sentiment Analysis and Opinion Mining,” Morgan & Claypool Publishers, USA, 2012.
- [13] A. Cambero, “A Comparative Study of Twitter Sentiment Analysis Methods for Live Applications,” 2016.
- [14] B. Liu, Sentiment Analysis and Opinion Mining., California: Morgan & Claypool Publishers, 2012.
- [15] P. Turney, Thumbs Up. Thumbs Down. Semantic Orientation Applied To Unsupervised Classification of Reviews, Philadelphia, 2002, pp. 417-424.
- [16] J. W. T. a. B. M. Wiebe, Identifying collocations for recognizing opinions, 2001.
- [17] C. L. L. T. J. J. L. Z. M. L. P. Tan, User-level sentiment analysis incorporating social networks., 2011.
- [18] A. B. S. a. G. : B. Das, Sentiment analysis: what is the end user’s requirement?, 2012, p. 35.
- [19] L. N. J. T. Y. C. K. Tan, Phrase-level sentiment polarity classification using rule-based typed dependencies and additional complex phrases consideration, 2012, pp. 650-666.

- [20] T. W. J. H. P. Wilson, Recognizing contextual polarity in phrase- level sentiment analysis, 2005, p. 347–354.
- [21] P. Tetlock, More than words: Quantifying language to measure firms' fundamentals., 2008, p. 1437–1467.
- [22] S. H. K. M. R. N. T. R. G. A. a. R. Blair Goldensohn, J. Building a sentiment summarizer for local service reviews., 2008, p. 14.
- [23] Z. M. A. L. B. H. M. C. M. a. G. R. Chen, Exploiting domain knowledge in aspect extraction, 2013, p. 1655–1667.
- [24] L. a. P. B. Lee, Opinion mining and sentiment analysis, Foundations and Trends in Information Retrieval, 2008, pp. 1-135.
- [25] G. E. J. F. A. H. T. N. R. & D. D. Miner, Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications, 2012.
- [26] W. H. A. & K. H. Medhat, Sentiment Analysis algorithms and applications: A survey, AinShams Engineering Journal, 2014, pp. 1093-1113.
- [27] B. W. H. a. Y. Liu, Integrating classification and association rule mining., 1998.
- [28] O. Pietquin, A framework for unsupervised learning of dialogue strategies, 2014.
- [29] J. a. L. Yang, J. Temporal Variation in Online Media, 2011.
- [30] H. a. B. L. Minging, Mining and summarizing customer reviews, 2004, pp. 168-177.
- [31] S. H. E. Kim, Determining the sentiment of opinions, 2004.
- [32] W. G. Parrott, Emotions in social psychology: Volume overview., Philadelphia, 2001, pp. 1-19.
- [33] B. Liu, Sentiment Analysis and Opinion Mining, California: Morgan & Claypool Publishers, 2012.
- [34] W. H. A. & K. H. Medhat, Sentiment Analysis algorithms and applications: A survey, 2014, pp. 1093-1113.
- [35] J. M. Nazzal, I. M. El-Emary and S. A. Najim, "Multilayer Perceptron Neural Network (MLPs) For Analyzing the Properties of Jordan Oil Shale," IDOSI Publications, King Saudi Arabia, 2008.
- [36] J. Yosinski, J. Clune and Y. Bengio, "How transferable are features in deep neural networks?," Cornell University.
- [37] R. Kishore and T. Kaur, "Backpropagation Algorithm: An Artificial Neural Network Approach for Pattern Recognition," 2012.
- [38] G. F. D. Duff and D. Naylor, "Differential Equations of Applied Mathematics," 1996.
- [39] A. Karpathy, The Unreasonable Effectiveness of Recurrent Neural Networks, 2015.
- [40] Colah, Understanding LSTM Networks, 2015.
- [41] E. Loper and S. Bird, "NLTK: The Natural Language Toolkit," University of Pennsylvania, Philadelphia.
- [42] L. Zhao and C. Zeng, "Using Neural Networks to Predict Emoji Usage from Twitter Data," CA.
- [43] R. T.-W. Lo, B. He and I. Ounis, "Automatically Building a Stopword List for an Information Retrieval System," Glasgow, UK.

- [44] V. Balakrishnan and E. Lloyd-Yemoh, “Stemming and Lemmatization: A Comparison of Retrieval Performances,” 2014.
- [45] O. Popescu and N. P. A. Vo, “Fast and Accurate Misspelling Correction in Large Corpora,” University of Trento, Trento, Italy.
- [46] J. Pennington, R. Socher and C. D. Manning, “Global Vectors for Word Representation,” Stanford, CA, 2014.
- [47] D. Mahajan, R. Patel and V. Sanker, “Word2Vec using Character n-grams,” CA.
- [48] N. Srivastava, G. Hinton and A. Krizhevsky, “A Simple Way to Prevent Neural Networks from Overfitting,” University of Toronto, Toronto, Ontario, 2014.
- [49] S. Ioffe and S. Ioffe, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” Google Inc., USA.
- [50] J. Collis, “Glossary of Deep Learning: Batch Normalisation,” Jun 27.
- [51] Rui and Shu, “A Gentle guide to using batch normalization,” 2016.
- [52] N. E. West and T. J. O’Shea, “Deep Architectures for Modulation Recognition,” U.S. Naval Research Laboratory and Virginia Polytechnic Institute and State University, Washington, D.C.
- [53] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” 2012. 437-478.
- [54] D. Richard and E. Turner, “Convolutional neural networks for computer vision,” 2014.
- [55] S. Lawrence, L. Giles and A. C. Tsoi, “What Size Neural Network Gives Optimal Generalization? Convergence Properties of Backpropagation,” University of Maryland, 1996.
- [56] P. Goyal, P. Dollar, R. Girshick, P. Noordhuis and A. Tulloch, “Accurate, Large Minibatch SGD,” Facebook.
- [57] P. Baldi and P. Sadowski, “The Dropout Learning Algorithm,” University of California, Irvine.