

Russian Federation Ministry of Education

Moscow Institute of Physics and Technology (State University)

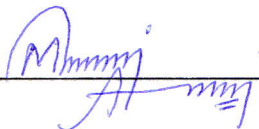
Radiotechnique and Cybernetics Department


Intelligent Information Systems and Technology Chair

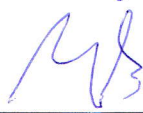
## **Text classification with Deep Learning Neural Networks**

Graduation qualification work  
(Master thesis)

Direction of training: 01.04.02 Applied mathematics and informatics

Performer:  
Student:  / Maaz Amjad /

Supervisor:  
Dr., prof.  / Avedian E.D /

Consultant:  / Ilya Voronkov /

Moscow 2017

## Review

---

For the master's thesis of a student of 193D group entitled by

### “Text Classification with Deep Learning Neural Networks”

The master's thesis devoted to solve an important problem in natural language processing for classification of texts which has various applications in practical life.

In inception, first two chapters has focussed on the review of different techniques of machine learning and deep learning that are being used to solve many classification problems. The most prominent and significant methods are identified and their modelling is carried out. The central goal of this thesis is to classify semantics of the texts. Most research involving experimental setup, IMDB dataset is used and various methods were implemented on the basis of both traditional machine learning as well as deep learning (neural networks) methods are investigated and their features. For experiments, the classifiers like Naïve Bayes, Multilayer Neural Network, Convolution Neural Network and Long Short Term Memory (LSTM) models are used. In addition, it may therefore be advantageous to examine the comparative review of the implemented methods, their strengths and the weakness were indicated in results. The above studies provide valuable information regarding the classification tasks. Therefore, future investigation using the LSTM would be helpful to achieve better accuracy.

This current thesis work is of practical and theoretical interest which is carried out at sufficiently very good level. It also meets all the requirements required for the final qualification of the master's thesis, deserves “excellent” (10) marks and awarding him qualification of “Masters of Science” in the direction of training: 01.04.02. “Applied mathematics and informatics”.

Review, Doctor of Technical Sciences,

Prof. Eduard Avedian

Moscow Institute of Physics and Technology, Russia

*Ally - 13.06.2017*

## Review

---

For the master's thesis of a student of 193D group entitled by

### “Text Classification with Deep Learning Neural Networks”

The master's thesis devoted to solve an important problem in natural language processing for text classification which has various applications in practical life.

To begin with, the main focus of first two chapters is the literature review of different techniques that are being applied to solve many classification tasks. The focus of a literature review, however, is to summarize and synthesize the arguments and ideas of different models without adding new contributions. The most prominent and significant methods are identified and their modelling is carried out. The advantages and disadvantages of each method are presented. In experiment setup, various methods were implemented on the basis of both traditional machine learning as well as deep learning (neural networks) methods. In addition, a comparative review of the implemented methods, their strengths and the weakness were indicated in results.

This thesis work is of practical and theoretical interest which is carried out at sufficiently very high level. It also meets all the requirements required for graduate work of masters, and deserves to be assessed “excellent” (10) marks. The author of the final qualifying work Maaz Amjad can be awarding the qualification “Masters of Science” in the direction of training: 01.04.02. “Applied mathematics and informatics”.



13.06.2017

Review, Candidate of Technical Sciences,

Prof. Georgy Ostapenko

Moscow Institute of Physics and Technology, Russia

© Copyright by Maaz Amjad, 2017.

All rights reserved.

# Abstract

One key problem with text classification learning algorithms is that they require many hand-labeled examples to learn accurately. This thesis presents an investigation into the extent to which review texts can be classified according to positive and negative labels. It also presents that supervised learning algorithms that utilize a small number of predesigned (labeled) examples can produce high-accuracy text classifiers.

To this purpose, different distinct classification approaches are employed for text categorization; Nave Bayes, Multilayer neural network, Convolution neural network and Recurrent neural network (LSTM). The results show that the approach of traditional machine learning methods has no positive effect on classification accuracy. In addition, regarding classification of sentiment labels, our work presents experiments for each classifier. The best prediction results for Nave Bayes, Multilayer neural network, Convolution neural network and Recurrent neural network (LSTM) are very similar. The best performances of each classifier do not differ significantly from each other, whereas all scores significantly outperform the baseline scores. When classifying sentences (sequence), the best results for the LSTM approach are significantly prominent, whereas other classifiers achieve the good results as well.

# Acknowledgements

I would like to take this opportunity to thank the people whose joint efforts assisted me in writing this thesis. First and foremost, my greatest thanks go to Prof. Alexander Galushkin(late), Prof. Eduard Avedian, Prof. Alexey Nazarov, Ilya Voronkov, Dima Pantyukhin for introducing me to the Wonderful world of artificial intelligence specially heartiest thanks to Ilya Voronkov for introducing me Natural Language Processing, and for supervising this research. Their constant support, providing invaluable knowledge and encouragement, insight, and a valuable big-picture perspective thorough guidance, and great patience enabled this work.

I especially want to express my gratitude to Prof. Andrei Raigorodskii, Dr. Alex Dainiak, Prof. Roman Karasev and Prof. V. L. Dol'nikov for their assistance beyond the research processes throughout various academic tasks required along the way.

This thesis would not have been possible without the love and support of my parents and my family. They were my first teachers and inspired in me a love of learning and a natural curiosity by encouraging me to pursue my academic goals and dreams.

Last and most importantly, I lovingly thank MIPT international office staff specially Irina Obukhova, Polina Golubkova, Marina Fomina, Ksenia Nikitina, Evgenia Lebedeva and to my friend Artur Kryzhko for providing support to endure and enjoy it all.

At the end of the day, the most overwhelming key to a my success is the positive involvement of my lovely parents.

Dedicated To

My Parents

# Contents

Abstract . . . . .	iii
Acknowledgements . . . . .	iv
List of Tables . . . . .	viii
List of Figures . . . . .	ix
<b>1 Introduction</b>	<b>1</b>
1.1 Automatic text classification . . . . .	2
1.2 Scope . . . . .	3
1.3 Main Contributions . . . . .	4
1.4 Methodology . . . . .	6
1.5 Organization of the document . . . . .	7
<b>2 State of the Art</b>	<b>8</b>
2.1 Text classification an interesting problem . . . . .	8
2.2 How to measure Text classification . . . . .	10
2.3 Machine Learning for text classification . . . . .	11
2.4 Methods of Text representation . . . . .	11
2.4.1 TF-IDF . . . . .	11
2.4.2 Vector representation of words . . . . .	13
2.4.3 Bayesian classifier . . . . .	16
2.5 Text Classification Approaches . . . . .	18



2.5.1	Naive Bayes . . . . .	18
2.5.2	Support Vector Machines . . . . .	19
2.5.3	k-nearest neighbors algorithm . . . . .	23
2.6	The hidden Markov model . . . . .	24
2.6.1	The Markov model . . . . .	24
2.6.2	The Hidden Markov Models (HMMs) . . . . .	27
2.7	Multilayer Neural Network . . . . .	27
2.8	The method of backpropagation error . . . . .	30
2.9	Convolutional Neural Networks for text classification . . . . .	32
2.10	Recurrent Neural Networks for text classification . . . . .	34
<b>3</b>	<b>Experimental Setup</b>	<b>36</b>
3.1	Data collection and Data Preparation . . . . .	36
3.2	Naive Bayes . . . . .	38
3.3	tf*idf Sequence Classifier . . . . .	40
3.4	Multilayer Neural Network . . . . .	41
3.5	Convolutional Neural Network . . . . .	42
3.6	LSTM For Sequence Classification . . . . .	44
3.7	LSTM and CNN For Sequence Classification . . . . .	46
<b>4</b>	<b>Results</b>	<b>47</b>
4.1	Naive Bayes (multiple entry on text) . . . . .	47
4.2	Multilayer Neural Network . . . . .	48
4.3	Convolutional Neural Network . . . . .	49
4.4	LSTM For Sequence Classification . . . . .	51
4.5	LSTM and CNN For Sequence Classification . . . . .	52
4.6	Conclusion . . . . .	53
	<b>Bibliography</b>	<b>55</b>

# List of Tables

3.1	Number of documents per category, classification based on . . . . .	37
-----	---	----

# List of Figures

2.1	Word Embeddings . . . . .	14
2.2	CBOW-Skip-gram . . . . .	15
2.3	Word2vec . . . . .	15
2.4	Support Vector Machines . . . . .	20
2.5	KNN method . . . . .	23
2.6	The Hidden Markov Model . . . . .	27
2.7	Signal-flow graph of the perceptron . . . . .	28
2.8	Multilayer Neural Network . . . . .	29
2.9	backpropagation . . . . .	31
2.10	vector representation of words and convolution . . . . .	33
2.11	Recurrent Neural Network for classification . . . . .	34
3.1	Review Length in Words for IMDB Dataset . . . . .	37
4.1	Multilayer Neural Network . . . . .	49
4.2	Convolutional Neural Network . . . . .	50
4.3	Recurrent Neural Network . . . . .	51
4.4	LSTM with dropout . . . . .	51
4.5	LSTM and Convolutional Neural Network . . . . .	52

# Chapter 1

## Introduction

A text can have various categorises depending on their contexts as think about we work for a site that maintains a general list of new jobs for indigenous people from many certain resources. A user of the internet site might find new career opportunities by surfing around all openings in a specific job category. On the other hand, these job postings are spidered from the Web, and do not come with any category label. It would be so helpful to have a system that not only investigates the text automatically but also have the capability to make the decision itself instead of reading particularly each job post to ascertain the label, This kind of computerized process is known as *text classification*. Frequently these text (document) consist of many categories, such as newspapers, magazines and research papers etc. In supervised learning, text classification systems categorize documents into one (or several) of predefined sets of categories, based on their textual content.

This chapter answers the following questions:

- What is Text classification about?
- What is the scope of our work?
- What were our contributions?

- Why are they important?
- How did we reach these contributions?

## 1.1 Automatic text classification

Classification in data mining and machine learning is a technique used to predict what categorical class a data illustration belongs to. Text classification has great practical importance today given the large volume of on-line text available. In recent years there has been an explosion of electronic text from the Globe Wide Web, such as corporate databases, electronic E-mails, digital libraries systems, and more broadly chat rooms. On the other hand, one way of organizing this immense amount of data is to classify it into descriptive or topical taxonomies. There are numerous text documents available in electronic sort and most of the data are becoming available perpetually. The Web itself contains over a billion records as millions of people send an e-mail every day.

Text classification is important because it is a key task to other natural language processing tools such as:

- **Categories to free-text:** To predict what categorical class a data illustration belongs to.
- **Information retrieval:** It plays an important role for understanding between humans and machines of human language to extract information from unstructured text(document).
- **Question Answering:** The chat-bots provide giant help to customers get right to the point without the wait by answering customer questions.

- **Knowledge Acquisition:** “John was walking along the direction of North Bank”. The land beside a river or a name of the financial district in Florida, USA?
- **Real world applications:** It is used in customer service, especially in banking services, retail, and hospitality.
- **Corpus categorization:** This is very important tool to create and evaluate a system that assist you in classifying a document into several categorize.

Discussion about quality is important because one of our findings transforms some classification systems into high quality/medium coverage systems.

## 1.2 Scope

Text classification methods usually work by analyzing and extracting information from different knowledge resources and scoring the senses with these resources through different classes. Classification methods can differ in the way they score a sense, the way they select its target class, and the way they load the knowledge resources.

There are some problems with the unsupervised text classification methods. The first problem is that not all words of an unstructured text are used for determining the desired output because only keywords are recognized. Therefore, sometimes obvious output are not suggested, therefore, unwanted suggestions are given sometimes. The second problem is that how the system will learn from its errors. The third problem is that the suggestions are currently not ordered, it would be so difficult to decide the correct output if many outputs are suggested. Deep learning techniques enable us to find out the solutions of these problems. Supervised text classification could solve these problems. The first problem is solved since the complete text is used for both training data and new texts that need classification. The second problem can

be solved by expanding the training data. By using supervised text classification technique, ordering could be easily implemented on the text, since most of the supervised text classification techniques are able to calculate the relevance of an output to the unlabeled text. So, supervised techniques are a lot clearer and more useful for classification algorithms.

This research describes some novel improvements that are usable with several classification methods. These improvements add some changes to the target class selection and knowledge resources loading procedures.

## 1.3 Main Contributions

The thesis has one central research question. What is the best classification method for (Review) text classification? The goal of this question is finding a model that describes the best way of classifying reviews of business text(document). In order to answer the main research question, four questions are defined as follows.

- **What are the characteristic features that complicate business text classification? Can we use unlabeled data to perform text classification greatly?**

There are certain types of models to use for text classification that are simple in use in comparison to the complexity of human authoring. They maintain no sense of syntax, context, or even word order. It's certainly not apparent that maximizing the probability of an imperfect model raises distinction accuracy of classification. Will our models be enough to represent for the purposes of text classification task? There are some possible reasons to be at initially pessimistic. Similar proposed model approaches for related textual content tasks, such as part-of-speech tagging [19] [5] and information removal [23] have revealed that by adding unlabeled data into supervised learning caused to decreases the per-

formance of these systems. We exhibit that proposed models are very helpful and valid approach for review classification by using unlabeled data.

- **What are the restrictions of a keyword scanning system for business text classification? Can we use proposed model for all types of data set for text classification?**

Distinct text classification domains are very substantially same in respect to their properties. Some of them consist of short documents and some of them contain massive lengths. Several projects are very easy to understand and some are hard due to its complexities. Some classes are narrow in nature while others are multi-faceted. With such a huge amount of text, it is worthwhile to consider that whether all tasks can be done successfully by using unlabeled data for proposed model. It is demonstrated that for some domains, it is easy to use unlabeled data clearly with suggested model. But for some other tasks, dealing with unlabeled data with the proposed model drops the classification accuracy.

- **What are other distinct classification methods for business text classification? Can we improve accuracy by using labeled data and sparse data set?**

When proposed model needs no adaptation unlabeled data to improve accuracy significantly, but even they cannot compensate for an extreme lack of labeled data. In these cases, the expected maximization optimization process is not able to find highly probable Parameters. Viewing as expected maximization an iterative hill-climbing algorithm, the initialization given by sparse labeled data are the source of the trouble. This weakness can be confront in two ways: by choosing documents for labeling that provide high-quality initializations, and



second by adopting other likelihood maximization techniques that are insensitive to poor initialization.

- **For given the best classification method, what fluctuations of this classification method performs best? Can we use unlabeled data for more Challenging text classification domains?**

To achieve nice results by using unlabeled data for suggested approach, the most important fact is that classification precision and model likelihood should be correlated. This condition is provided mostly for perfect model and sufficient unlabeled data, but there are no guarantees for simple text models. It is difficult to get good classified output if our model assumptions are unable to approximate data distribution correctly. In these cases, it might be possible to make some changes in our assumptions to get more closely data. Suggested models are used in two different directions, one is by modeling the sub-topic structure of a class, and other check the super-topic hierarchical class relationships. These adapted models are helpful in training of text classifiers and allow to use unlabeled data in useful manners for that harder domain.

The answers for these above questions is the crux to the main research question. The characteristics features of business texts and the limitations of a keyword scanning system have a great impact on the destination of viewing classification methods, other than keyword scanning. When the best classification method is obtained, it can be further modified in order to achieve the best classification results.

## 1.4 Methodology

This thesis formalize on getting the best results for automatic text classification tasks. It is consisted of two main parts. In the first part, the limitations of the conventional methods are described. In the second part, a novel approach will be implemented

in order to perform supervised text classification tasks and then results would be evaluated.

## 1.5 Organization of the document

This thesis is organized as follows.

- **Chapter 2** contains basic knowledge for understanding text classification and the current methods that are being used.
- **Chapter 3** contains the description of methods and practices used for constructing our implementation. Also, it contains the description of the classification methods used being improved by our proposed methods.
- **Chapter 4** contains results of our implementation and concludes this thesis.

# Chapter 2

## State of the Art

Text classification is an open problem. There are many approaches that try to accomplish it.

This chapter answers the following questions:

- Why is Text classification an interesting problem?
- How to measure Text classification?
- What are supervised Text classification methods and how do they work?
- What is the importance of
  - Multilayer neural networks?
  - Convolutional neural networks?
  - Recurrent neural networks?

### 2.1 Text classification an interesting problem

In Machine learning, Machine learning algorithms frequently employ classifiers, for example, logistic regression (LR), support vector machine (SVM) and naive Bayes

(NB) classifiers etc. Nonetheless, these techniques contain the issues of data sparsity problem. Text classification is a classic topic for natural language processing and has many important applications in topics such as parsing, semantic analysis, information extraction and web searching [1]. Therefore, it has become a source of attraction for many researchers.

Natural language processing is an interesting topic for three main reasons such as philosophical, applications, learning. In the present time, we as human have defined ourselves more in terms of our ability to speak and understand to each other. This ability to use language is something that we feel sets us apart from all other animals and machines. Secondly, we really would like to be able to talk with our computers and used them for various purposes. There is no doubt that clicking is a right thing to do in but talking is natural, and we want to be able to communicate with our machines. Thirdly, we want that our computers to be smarter, and much of human knowledge is written in the form of sentences and paragraphs of text but not written in the formal database or informal procedures written in codes. It means that most of the information is in text and if we want our computers to be smart, they would be able to read the text and make sense of it.

Natural language processing has many applications in artificial intelligence. One application is text classification. Text classification is also called text categorization. It focuses on classifying pictures, text documents, and music pieces in one or more categories (also called classes or labels). The categories could be the title, a theme of the text, year of publication, publication organizations etc.

Text classification assigns a text document (a document could consist of one word, one sentence or an entire paragraph), to one or more predefined classes. This kind of task can be done by hand or automatically. To sort out a large number of text documents manually is an expensive and time-taking task. Classifying documents automatically can be done by using keyword scanning or by labeling training data

set. For manual text classification, this might not be the case, since each person expresses a text in a different way. Automatic text classification can be divided into three categories; supervised, unsupervised and semi-supervised. Supervised text classification uses labeled training data to categorize texts. The texts will, therefore, be classified into different categories which are predefined in the training data. Unsupervised text classification does not rely on labeled training data set. One example of an unsupervised text message classification system is a rule-based text classification system, such as keyword scanning. Semi-supervised text classification is a mixture of supervised and unsupervised text classification. To classification such type of tasks, both labeled and unlabeled training data sets are used.

## 2.2 How to measure Text classification

Suppose that, a document  $d \in \mathcal{D}$  of a text is provided, where  $\mathcal{D}$  is the document space; and  $\mathcal{C} = \{c_1, c_2, \dots, c_k\}$  is a fixed set of classes having length  $k$ . In text classification, classes are defined as categories or labels in a document space. In general, the document space  $\mathcal{D}$  is a certain kind of high-dimensional space, and the classes are defined by human intervention for application purposes. For example,  $\langle d, c \rangle = \langle \text{Moscow is no longer most expensive cities in the world, Russia} \rangle$

as in the example Russia and documents that talk about expensive cities. We are given a training set  $\mathcal{T}$  of labeled documents  $\langle d, c \rangle$ , where  $\langle d, c \rangle \in \mathcal{D} \times \mathcal{C}$ . Our central goal is to achieve a classification function  $\psi$  to classify our document, which transforms documents to certain classes by using learning algorithm or method.

$$\psi : \mathcal{D} \rightarrow \mathcal{C}$$

This kind of learning process is called supervised learning because we do train our model by using pre-categorized document having classes and labels. We denote the

supervised learning method by  $\chi$  and define  $\chi(\mathcal{T}) = \psi$ . This learning method  $\chi$  takes the training set  $\mathcal{T}$  as an input and returns the learned classification function  $\psi$ .

## 2.3 Machine Learning for text classification

State-of-the-art approaches are commonly classified into two classes: supervised (machine learning) and unsupervised. Classification is the problem of selecting a category for a new observation. Classification problem has the following elements:

- **Categories.** Categories are the possible classes in which an observation should be assigned to a class. For instance, when classifying sentiments (reviews) you will have two classes the positive and negative classes and classes are pre-defined.
- **Features.** Features are the crucial values used for determining is an observation belongs to a certain class or different class. For instance, when classifying sentiments (reviews) you can use certain words or phrases to recognize like this movie is amazing deciding a new observation belongs to which class.
- **Training data.** The training corpus is a set of pre-defined examples used for the classifier learning how to classify new instances.

In text classification, the classes are the categorises extracted from the dictionary; the features are words in the context; and, the training data is a manually tagged corpus such as IMDB.

## 2.4 Methods of Text representation

### 2.4.1 TF-IDF

The Sequence Classifier is a supervised learning algorithm which tries to look for similarities between documents based on the idea that the importance of a pattern

in a document can be weighted using the statistics of occurrence of that pattern [2]. These individual weights can then be used to score the entire document, and to determine which document belongs to which class. This concept originated in the field of Information Retrieval, where the relevance of a document to a given search query is measured by applying the commonly used TF\*IDF metric.

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure. It consists of two terms TF \* IDF. TF simply measures the frequency of occurrence of a term in a document  $d$  (when applied to text classification, the number of times the term occurs in a class  $c$ ). The TF weight in itself is not a very useful approach for determining relevance between a term and a class, as very common terms will automatically have very high TF values, even though such terms are not useful for discriminating between classes. Therefore, the second factor in the TF \* IDF metric is the inverse document frequency, or IDF, which is a measure for how rare a term is in the entire document collection (how much information the word provides). In the classical case, for the term  $t$ , the frequencies TF and IDF are defined as follows:

$$tf_{t_i,d} = \frac{n_i}{\sum_k n_k} \quad (2.1)$$

where  $n_i$  is the number of occurrence of the term  $t_i$  in the document, and denominator represent the number of documents in which term  $t$  appears (total length of the document), which will be high for rare terms (i.e. words that occur in only a few documents), but low for common terms (i.e. words that appear in almost every document). In the classical case, Given a document collection  $D$ , TF-the density entry of the term in the document.

$$idf_{t_i,D} = \log \frac{|D|}{|d_i \subset t_i|} \quad (2.2)$$

where  $|D|$ -the total number of documents(Corpus), and the absolute symbol shows the number of document containing the term  $t_i$ .

Combining these two factors permits for the identification of terms that are most crucial for discriminating between classes. The reason is that these terms occur frequently in the given class (term frequency) but not frequent in other classes (inverse document frequency). This metric can now be utilized to compute a weight for each word in the training data with respect to each class. These weights can then be used to calculate a scores for a new document  $d$ , by adding the weight of each word in that document. It exhibits that if a word occurs less frequently, it will have high IDF frequency.

$$tf - idf_{t_i,d,D} = tf_{t_i,d} * idf_{t_i,D} \quad (2.3)$$

### 2.4.2 Vector representation of words

Word representations are a very important part of many natural language processing systems. The traditional approach is to represent words as indices in a lexicon, but this fails to outperform in order to capture the rich relational structure of the vocabulary. New approaches to representing words are vector-based models that outperform and do much better in this regard. These models encode continuous similarities between words in a lexicon as distance or angle between word vectors in a high-dimensional space.

Vector representation of words is a method in which each word is a numerical vector of  $R^n$ . The closer the semantic meanings of words, the greater the cosine proximity between word and vectors(the scalar vectors product). This method is widely used for solving text processing problems, including classification.

The method of the vector representation of words word2vec was proposed in [20]. Word2vec is a set of algorithms used for learning vector representations of words,





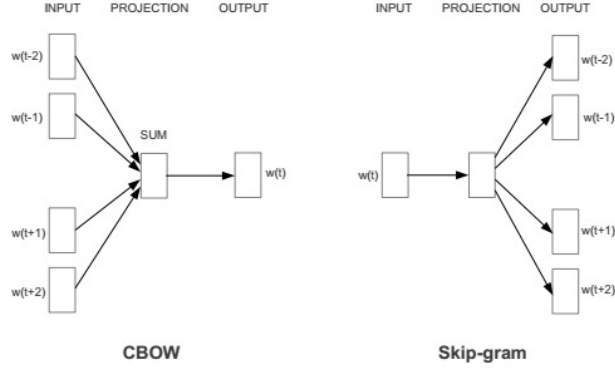


Figure 2.2: CBOW-Skip-gram

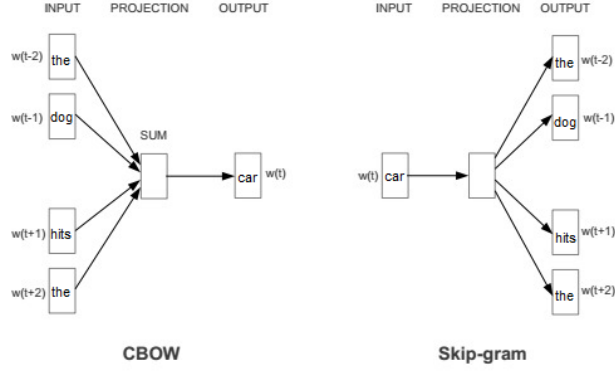


Figure 2.3: Word2vec

where  $Q_\theta(D = 1|w_t, h)$  is the binary logistic regression probability. Under this probability, the model of seeing the word  $w$  in the context  $h$  in the dataset  $D$ , calculated in terms of the learned embedding vectors  $\theta$ . When the model assigns highest probabilities to the original words and least probabilities to noise words. Consequently, this objective is maximized.

In [21] another method (GloVe) of vector representation of words has been proposed. It takes probability of a single word in the context that is different from other words in the document.

The cost function for the GloVe model:

$$J = \sum_{i,j=1}^V f(X_{i,j})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij}))^2 \quad (2.6)$$

where

$$f(x) = \begin{cases} (\frac{x}{x_{max}})^\alpha & : \text{if } x \leq x_{max} \\ 1 & : \text{otherwise} \end{cases}$$

The correlation (continuous similarities) between two words can be measured as the cosine distance between two vectors (scalar product of vectors):

$$Cos(\theta) = \frac{A.B}{\|A\|\|B\|} = \frac{\sum_{n=1}^{\infty} A_i \times B_i}{\sqrt{\sum_{n=1}^{\infty} A_i^2} \sqrt{\sum_{n=1}^{\infty} B_i^2}} \quad (2.7)$$

### 2.4.3 Bayesian classifier

The Bayesian classifier is a wide class of classification algorithms based on a maximum posterior probability. A posterior probability of an event is the conditional probability that is assigned after relevant observations. For classification, the likelihood function of each class is calculated by posterior probability. Object (text) refer to the class which has maximum posterior probability.

The Bayesian approach is based on Bayes' theorem which states that the probability of an event, based on prior knowledge of conditions that might be related to the event. It is one of the oldest approach to classification that still retain a strong position in the field of recognition. Thomas Bayes underlies many fairly successful classification algorithms and Naive Bayes is one of those which based on the same formula but with the additional assumption that object are described by independent sign. The assumption of independence greatly simplifies problem since it is much easier to estimate  $n$  one-dimensional densities than one  $n$ -dimensional density. A naive Bayesian classifier can be either a parametric or non-parametric depending on which method restored one-dimensional densities.

In[6] a naive Bayesian approach to the text classification is introduced. Each document can be considered as a set of words. The probability that a word  $w$  refers to the class  $c$  can be written as:

The probability of class  $c$ :

$$P(c) = \frac{n_c}{N}$$

where  $n_c$  is the number of classes of documents and  $N$  is the total number of documents in the training sample.

$$P(w|c) = \frac{1 + \sum_{d \in D_c} n_{wd}}{K + \sum_{w'} \sum_{d \in D_c} n_{w'd}} \quad (2.8)$$

where  $D_c$  is set of a class of documents in the training set whereas  $K$  is the size of dictionary.

The probability that this document  $d$  belongs to the class  $c$ :

$$P(c|d) = P(c) \prod_{w \in d} P(w|c)^{n_{wd}} \quad (2.9)$$

where  $n_{wd}$  is how many times the word  $w$  appears in the document  $d$ ,  $P(c)$  is the prior probability of class  $c$ , and  $(d)$  is a constant. To avoid calculation error, so use log in practice.

Probability:

$$P_{\log}(c) = \log \frac{n_c}{N} \quad (2.10)$$

$$P_{\log}(w|c) = \log \left( \frac{1 + \sum_{d \in D_c} n_{wd}}{K + \sum_{w'} \sum_{d \in D_c} n_{w'd}} \right) \quad (2.11)$$

$$P_{\log}(c|d) = \log \left( P(c) \prod_{w \in d} P(w|c)^{n_{wd}} \right) = P_{\log}(c) \sum_{w \in d} n_{wd} P_{\log}(w|c) \quad (2.12)$$

The Bayesian classifier uses the maximum posteriori (Maximum a posterior estimation) to determine the most likely class for a new document  $d$ :

$$c_d = \arg \max_{c' \in C} P_{\log}(c'|d) \quad (2.13)$$

On one hand, the Bayesian classifier works very quickly, its principal of operation is very simple and easy to use. On the other hand, it does not take into account order and semantic meanings of words in the text. So, its not suitable for classification task that requires the understanding of texts.

## 2.5 Text Classification Approaches

Text Classification can be attained in certain ways, the aim is always the same - to discover the classifier that describes how to assign a new document to a fixed number of classes. This classifier can be achieved or at least approximated by providing a supervised learning algorithm with an example collection of pre-classified documents.

### 2.5.1 Naive Bayes

The first technique employed in this research, Nave Bayes, is a notable and frequently used learning technique, which exploits a powerful but comparatively easy approach for learning the relationship between words and classes. The Nave Bayes method is based on the assumption that for a document  $d \in \mathcal{D}$ , choosing the best class  $c$  out of a set of all classes  $\mathcal{C}$  simply relies on choosing the most plausible class for that document[8]. This selection depends on the words that appear in each document. By keeping in mind that in order to attain naively suppose that the probability of words appearance in a document is not dependent on the context and order of the words (e.g. it would not be imaginable to realize the sentences “Ivan loves Maria and “Maria loves Ivan).

This premise yields the bag-of-words representation of a document, and despite the fact that this representation of words is not feasible in most real-world circumstances, the Naive Bayes technique frequently outperforms in most of the text classification tasks[18]. The probability that the label of a class  $c$  should be assigned to an observed document  $d$  is computed using the following rule:

$$P(c|d) \propto P(c) \prod_{i=1}^N P(w_i|c) \quad (2.14)$$

Where  $P(c)$  is the proportion of documents belonging to the class  $c$  (i.e. the prior probability of a class). This proportion can be visualized as a weight that represents the relative frequency of the class; most occurred classes are more likely to be the correct selection than less occurred classes. The probability is the number of times a word  $w_i$  appears in classified documents with class  $c$ , divided by the total number of terms in a classified document with class  $c$  (i.e. the likelihood that a word belongs to a class  $c$ ). In other words, it is a measure for how much evidence  $w_i$  contributes to the probability of the document belonging to class  $c$ [2]. In the end, the class with the maximum score can be assigned to a document, as this is the best (i.e. most probable) class depend on the terms in that document.

## 2.5.2 Support Vector Machines

Support Vector Machines is an algorithm used for classification and regression analysis. The main idea is that it constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space and maximize distance between parallel hyperplanes. The algorithm works on the assumption that the greater the difference or distance between the parallel hyperplanes, the smaller the standard error of the classifier.

Search optimal separating hyperplane[14]: Linear classifier is the best way to solve the problem. The idea is as follows, find a line that separates all the red points from

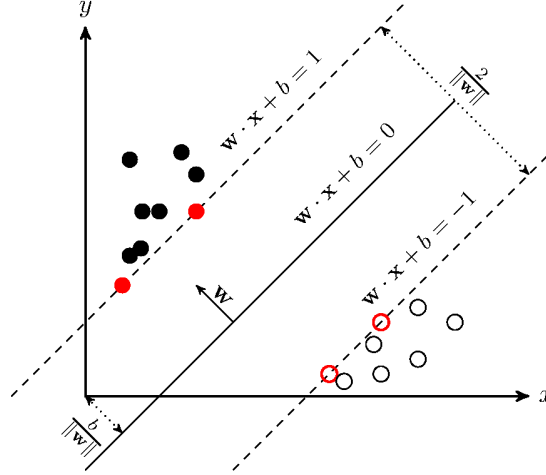


Figure 2.4: Support Vector Machines

the white points. If we manage to find a line, then classify each new point will be as follows: if the point lies above the line, it is black if it is below the white. Formalization of classification, the important thing is to find a vector  $w$  such that for a certain boundary values of  $b$  and a new point  $x_i$  is performed.

$$\begin{aligned} w \cdot x_i > b &\implies y_i = 1 \\ w \cdot x_i < b &\implies y_i = -1 \end{aligned} \tag{2.15}$$

Where  $w \cdot x_i = b$  describes a hyperplane that separates the classes in  $R^n$  space.

Since we are free to choose separating hyperplane, it is necessary to take advantage of it in order to improve classification of text. So, it is helpful to divide line so that it is as far distant from nearest points of both classes. i.e find such vector  $w$  and the number of  $b$  that for  $\epsilon > 0$  is satisfied:

$$\begin{aligned} w \cdot x_i > b + \epsilon &\implies y_i = 1 \\ w \cdot x_i < b - \epsilon &\implies y_i = -1 \end{aligned} \tag{2.16}$$

Classification algorithm will not change, if  $w$  and  $b$  simultaneously multiply at the same constant. By taking advantage of this and chose constant so that all the boarder (i.e closest to the dividing hyperplane) points following conditions:

$$w.x_i - b = y_i \quad (2.17)$$

This can be done, since the optimal position separating hyperplane contain boundary objects which are from the same distribution and the other objects are further away. Multiply then inequality to  $\frac{1}{\epsilon}$  and choose  $\epsilon=1$ . Thus, for all vectors  $x_i$  for training:

$$\begin{aligned} w.x_i - b &\geq 1 \quad : \text{if } y_i = 1 \\ w.x_i - b &\leq -1 \quad : \text{if } y_i = -1 \end{aligned} \quad (2.18)$$

where  $-1 < w.x_i - b < 1$  defines the dividing line between classes and the band width is  $\frac{1}{\|w\|}$ . The wider the band, the more confident you can be classified document respectively, which implies that the large band is the best fit.

Case linear separability: It is formulated the conditions of the search problem of optimum separating strip.

$$y_i(w.x_i - b \geq 1) \quad (2.19)$$

The goal is to find  $w$  and  $b$  to perform all linear restriction. Thus, as little was possible norm of the vector  $w$  (hence wider separating string) that is to be minimized:

$$\|w\|^2 = w.w \quad (2.20)$$

This problem is called quadratic optimization problem with linear constraints to find the minimum of the quadratic function.



Case absence linear separability: it allows algorithm to make mistakes in the training data. A new approach is introduced a set of additional variables  $\xi_i > 0$ , characterizing the magnitude of the error facilities for  $x_i \in [x_1 \dots x_n]$ . This makes it possible to mitigate limitations(inequalities):

$$y_i(w.x_i - b \geq 1) - \xi_i \quad (2.21)$$

Reformulate the problem of finding the optimal separating: in data limitations to minimize the amount of

$$\|w\|^2 + C \sum \xi_i \quad (2.22)$$

In order to solve this issue, Lagrange method can be used. Lagrange tried to reduce the conditional minimum of the search task(with restriction) to search task the absolute minimum(without limitation) then use standard methods of finding minimum functions. To do this task, it is necessary to make some changes in objective function (i.e the function that must be minimized).

In formulating the problem in terms of Lagrange model, it is crucial to find atleast of  $w$ ,  $b$ ,  $\xi_i$  and a maximum of  $\lambda_i$  function

$$\frac{1}{2}w.w + C \sum_i \xi_i - \sum_i \lambda_i(\xi_i + y_i(w.x_i - b) - 1) \quad (2.23)$$

where  $\xi_i, \lambda_i \geq 0$

In [7], this technique has been applied to a support vector classification texts. The advantages of support vector machines are as follows:

- A large number of text classes is suitable for support vector machine method because it is able to work with multi-dimensional data.
- Document vectors are sparse, may contain many zero entries.
- Most of the text classification tasks are linearly separable[12].

### 2.5.3 k-nearest neighbors algorithm

To the nearest neighbors(KNN) is one of the most used method of classification. The main principal of the method is that object belongs to the class that owns most of its neighbours. In other words, for an object, consider it to the nearest neighbors, then for each class, consider how many of these K neighbours are lie to this class, the object belongs to the class with the largest number neighbors [16].

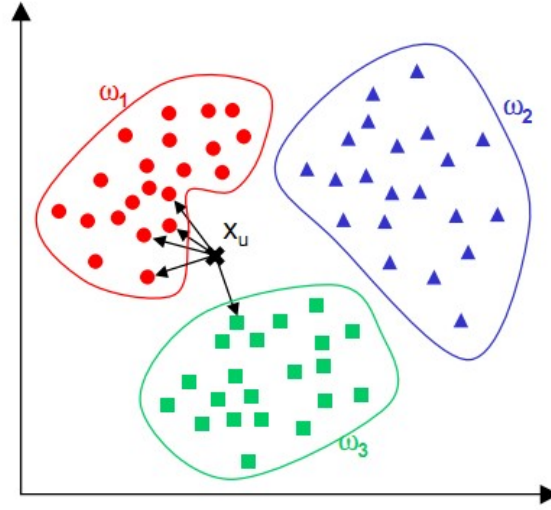


Figure 2.5: KNN method

KNN is widely used for classification of texts. To begin with, consider coordinates of texts in space. The size of the space is the number of terms in the body( the volume of the dictionary). Suppose TF-IDF for all texts in the package, we get representation of the text in the form of numerical vectors, each component vector-“ important” of the corresponding word for the given text. Coordinates texts are used to solve various problems, including classification[8].

In [24], KNN method was used to classify text. It has been shown that quality of classification of text using KNN is better than Bayesian classifier. The problem is that KNN is very slow speed in order to classify a new text. To perform classification for new text, it needs to calculate distant between this text and all texts in the corpus. It

is really difficult due to the fact that their numbers can be millions. In later section, a new approach has been proposed to do KNN optimization promptly and with greater speed.

## 2.6 The hidden Markov model

A Markov model is a stochastic model that is being used to model systems that is randomly changing with the assumption that future states does not depend on past but only depend on the current state. In other words, there is no past memory in a Markov process.

### 2.6.1 The Markov model

The simplest case is the Markov chain. The Markov Chain is a statistical model of a system that moves in a consecutive manner from one state to another. The observed value vector  $Z_t$  at time  $t$  depends only on the state at the previous point in time  $Z_{t-1}$  [22]:

$$P(z_t | z_{t-1}, z_{t-2}, z_{t-3}, \dots, z_1) = P(z_t | z_{t-1}) \quad (2.24)$$

The conditional distribution of the observed state with the state at the previous time does not change

$$P(z_t | z_{t-1}) = P(z_t | z_1); \quad t \in 2, \dots, T \quad (2.25)$$

Suppose that  $A_{ij}$  is observed transition probability  $z_t$  from the value  $i$  to the value  $j$  at any time  $t$ . The probability sequence  $\vec{z}$  is calculated as:

$$\begin{aligned}
P(\vec{z}) &= P(z_t, z_{t-1}, z_{t-2}, \dots, z_1; A) \\
&= P(z_t, z_{t-1}, z_{t-2}, \dots, z_1, z_0; A) \\
&= P(z_t | z_{t-1}, z_{t-2}, \dots, z_1; A) P(z_{t-1}, z_{t-2}, \dots, z_1; A) \dots P(z_1 | z_0; A) \\
&= P(z_t | z_{t-1}; A) P(z_{t-1}, z_{t-2}; A) \dots P(z_1 | z_0; A) \\
&= \prod_{t=1}^T P(z_t, z_{t-1}; A) \\
&= \prod_{t=1}^T A_{z_{t-1} z_t}
\end{aligned} \tag{2.26}$$

The logarithm of the likelihood function of the Markov model:

$$\begin{aligned}
\log P(\vec{z}; A) &= \log \prod_{t=1}^T A_{z_{t-1} z_t} \\
&= \sum_{t=1}^T \log A_{z_{t-1} z_t} \\
&= \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} \sum_{t=1}^T 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij}
\end{aligned} \tag{2.27}$$

The problem can be solved by using the method of Lagrange multipliers:

$$\begin{aligned}
&\max_A l(A) \\
&= \sum_{j=1}^{|S|} A_{ij} = 1, \quad i = 1, \dots, |S| \\
&A_{ij} \geq 0 : \quad i, j = 1, \dots, |S|
\end{aligned}$$

The Lagrangian is:

$$\mathcal{L}(A, \alpha) = \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} \sum_{t=1}^T 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij} + \sum_{i=1}^{|S|} (\alpha_i (1 - \sum_{j=1}^{|S|} A_{ij})) \quad (2.28)$$

Take partial derivatives and equate them to zero:

$$\begin{aligned} \frac{\partial \mathcal{L}(A, \alpha)}{\partial A_{ij}} &= \frac{\partial}{\partial A_{ij}} \left( \sum_{t=1}^T 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij} \right) + \frac{\partial}{\partial A_{ij}} \left( \alpha_i (1 - \sum_{j=1}^{|S|} A_{ij}) \right) \\ &= \frac{1}{A_{ij}} \sum_{t=1}^T 1\{z_{t-1} = s_i \wedge z_t = s_j\} - \alpha_i = 0 \end{aligned} \quad (2.29)$$

By simplification:

$$A_{ij} = \frac{1}{\alpha_i} \sum_{t=1}^T 1\{z_{t-1} = s_i \wedge z_t = s_j\} \quad (2.30)$$

Let's put  $A_{ij}$  back to Lagrangian and equate  $\alpha$  derivative to zero.

$$\begin{aligned} \frac{\partial \mathcal{L}(A, \beta)}{\partial \alpha_i} &= 1 - \sum_{j=1}^{|S|} A_{ij} \\ &= 1 - \sum_{j=1}^{|S|} \frac{1}{\alpha_i} 1\{z_{t-1} = s_i \wedge z_t = s_j\} - \alpha_i \end{aligned} \quad (2.31)$$

By simplification:

$$\begin{aligned} \alpha_i &= \sum_{j=1}^{|S|} \sum_{t=1}^T 1\{z_{t-1} = s_i \wedge z_t = s_j\} \\ &= \sum_{t=1}^T 1\{z_{t-1} = s_i\} \end{aligned} \quad (2.32)$$

By substituting value of  $\alpha$  in eq (2.30):

$$\hat{A}_{ij} = \frac{\sum_{t=1}^T 1\{z_{t-1} = s_i \wedge z_t = s_j\}}{\sum_{t=1}^T 1\{z_{t-1} = s_i\}} \quad (2.33)$$

### 2.6.2 The Hidden Markov Models (HMMs)

It is a Markov process with the assumption that there is doubt in what state the system is in at any given time. A hidden Markov model model is a probabilistic model of the sequence, which consists of a set of observed variables  $x = \{x_1, x_2, x_3, x_T\}$ ,  $x_t \in V = \{v_1, v_2, v_3, v_{|V|}\}$  and latent(hidden) variables  $z = \{z_1, z_2, z_3, z_{|T|}\}$ , and  $z_t \in S = \{s_1, s_2, s_3, s_{|S|}\}$ .

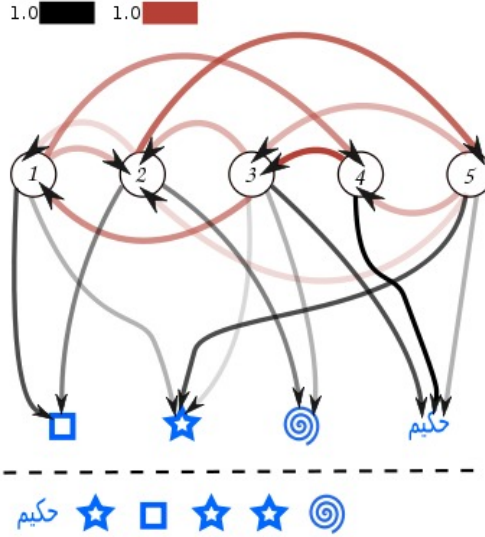


Figure 2.6: The Hidden Markov Model

## 2.7 Multilayer Neural Network

A multilayer perceptron (MLP), as well as a single-hidden-layer Multi-Layer Perceptron, is a feedforward neural network with one or more layers between input and

output layer. The word Feedforward means that data flows in one direction from input to output layer (forward) through some nonlinear activation function. This type of network is trained with the backpropagation learning algorithm despite the huge number of weighting coefficients. Mathematically, the output value of the neuron  $y_j$  through some nonlinear activation function  $\varphi$  can be written as

$$y_j = \varphi\left(\sum_{i=1}^n w_i x_i + b\right) = \varphi(w^T x + b) = \varphi(z_j) \quad (2.34)$$

where

$$z_j = \left(\sum_{i=1}^n w_{ij} x_i + b\right) \quad (2.35)$$

Here,  $\mathbf{w}$  denotes the vector of weights,  $\mathbf{x}$  is the vector of inputs,  $b$  is the bias value and  $\varphi$  is the activation function.

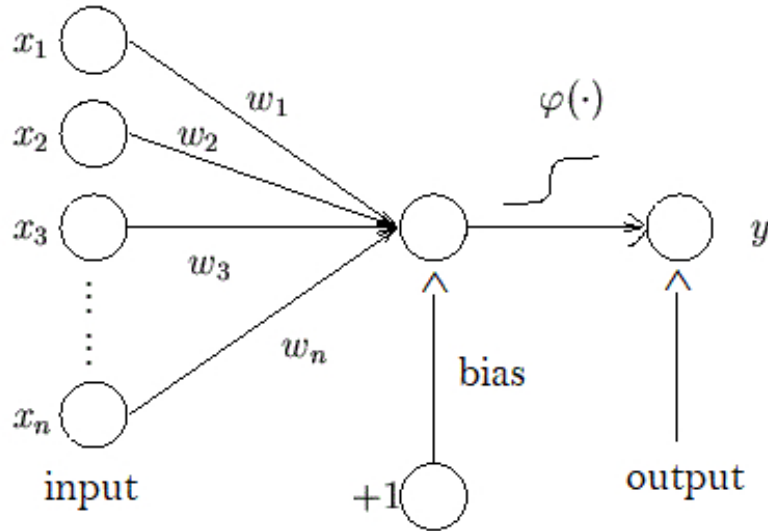


Figure 2.7: Signal-flow graph of the perceptron

An MLP (or Artificial Neural Network - ANN) with a multiple hidden layer can be represented graphically as follows:

Frequently used activation function and their derivatives.

Sigmoid:

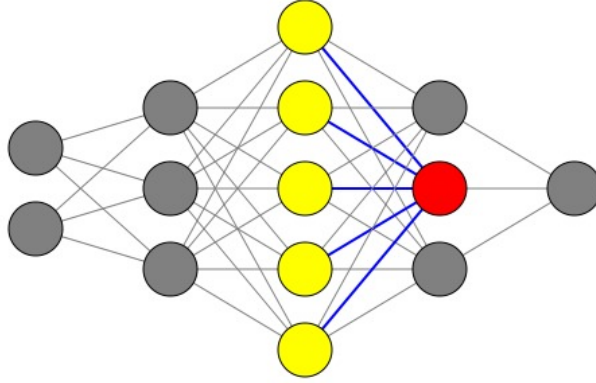


Figure 2.8: Multilayer Neural Network

$$\varphi(z) = \sigma(z) = (1 + e^{-z})^{-1} \quad (2.36)$$

$$\frac{\partial \varphi(z)}{\partial z} = \frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z)) \quad (2.37)$$

Tanh:

$$\varphi(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (2.38)$$

$$\frac{\partial \varphi(z)}{\partial z} = \frac{\partial \tanh(z)}{\partial z} = 1 - \tanh^2(z) \quad (2.39)$$

ReLU:



$$\varphi(z) = ReLU(z) = \max(0, z) \quad (2.40)$$

$$\frac{\partial \varphi(z)}{\partial z} = \frac{\partial ReLU(z)}{\partial z} = \begin{cases} 0 & : \text{ if } z \leq 0 \\ 1 & : \text{ if } z \geq 0 \end{cases} \quad (2.41)$$

## 2.8 The method of backpropagation error

Back-propagation is a detailed insight into how quickly changing the biases and weights changes the overall behavior of the neural network output to minimize the cost function. Suppose that  $\hat{y}$  is the target output vector of the neural network. Error in the output layer can be defined as:

$$\delta_i = y_i - \hat{y}_i \quad (2.42)$$

The basic idea of the backpropagation method is that errors in each hidden layer is calculated by the following: It should be noted that the current neural network is an effective classification method. It is fast and can be used to solve various problems. The learning process of neural networks require more resources and time to train the model. The learning process of neural networks is a crucial point to understand. It is carried out by comparison of the network's output(i.e., the outputs of the last layer in a network) with the corresponding teacher's instructions[2]. Architectures of different neural networks have different ways for network learning by back-propagation.

$$\delta_i = \sum w_{ij} \delta_j \quad (2.43)$$

For a fixed input vector  $x$ , a loss function of the network can be written as a functional of the mean squared error, which depends on the network weights:

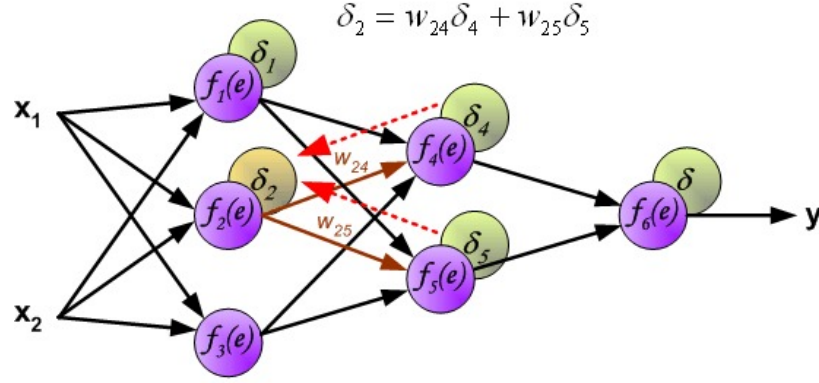


Figure 2.9: backpropagation

$$\xi_j(w) = \frac{1}{2} \delta_j^2 = \frac{1}{2} (y_j - \hat{y}_j)^2 \quad (2.44)$$

Taking partial derivatives of  $\xi$  weights:

$$\frac{\partial \xi_j}{\partial w_{ij}} = \frac{\partial \xi_j}{\partial y_j} \frac{\partial y_j}{\partial w_{ij}} \quad (2.45)$$

$$= \frac{\partial (\frac{1}{2} (y_j - \hat{y}_j)^2)}{\partial y_j} \frac{\partial \varphi(z_j)}{\partial w_{ij}} \quad (2.46)$$

$$= (y_j - \hat{y}_j) \frac{\partial \varphi(z_j)}{\partial z_j} \frac{\partial z_j}{\partial w_{ij}} \quad (2.47)$$

$$= \delta_j \frac{\partial \varphi(z_j)}{\partial z_j} \frac{\partial (\sum w_{ij} x_i)}{\partial w_{ij}} \quad (2.48)$$

$$= \delta_j \frac{\partial \varphi(z_j)}{\partial z_j} x_i \quad (2.49)$$

If the activation function of neural network is one of the listed functions 2.36, 2.38, 2.40, then the derivative  $\frac{\partial \varphi(z_j)}{\partial z_j}$  is calculated from the formulas 2.37, 2.39 and 2.41 respectively.

where  $x_i$  is the output of the previous neuron and  $\delta$  is the learning rate, which is meticulously chosen to ensure that the parameters (weights) converge to a response fast enough, without creating oscillations.

## 2.9 Convolutional Neural Networks for text classification

**Convolutional Neural Network** (CNN) is introduced in natural language processing in order to solve the biases issue. CNN has the ability to extract the semantic of texts in a very systematic way as compared to recurrent or recursive neural networks with time complexity  $O(n)$ . It can capture critical dialogues in a text by using a max-pooling layer. Although previous research on CNN reveals the kernel technique as a rigid window [3] [9]. In addition, it is so hard to find the most suitable size of a kernel (window size). If a size of the kernel is huge, it would be a cause of many parameter spaces (possibly hard to train it) whereas small size kernel may lead inaccurate results by missing discriminative information. Therefore, it leads to raising an important fact: would it be possible to get contextual information to a greater extent than traditional window-based neural networks and constitutes semantic of information in a more precise way to test classification tasks?

In [10], an approach has implemented using text classification methods vector representation of words and convolutional neural network. When text classification using a convolutional neural network, the following conditions are given.

- The filter has same width  $m$ , the matrix ( the input of a neural network whole words) come.
- It is possible to use simultaneously different filters to highlight various features in different sections.

According to [10], it can be concluded that the convolutional neural network works well for text classification, especially with word embedding method.

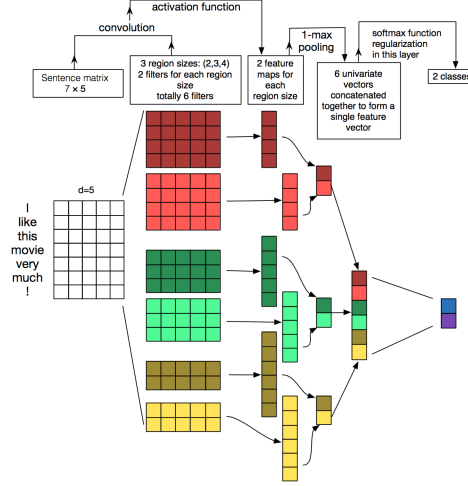


Figure 2.10: vector representation of words and convolution

Recurrent Convolutional Neural Network (RCNN)[13] to deal with restrictions of capturing important dialogues. In order to get nice results for learning word representation, a recurrent structure(bi-directional) is employed which may acquaint extensively less noise as a comparison with window based neural network. It also has the ability to extract appropriate contextual information as up till possible. In addition, when learning word representations, the model can grasp a bigger compass of word ordering. A maximum pooling layer that consequently judges which feature assume to be important to catch the key segment during text classification. By consolidating the recurrent structure and max-pooling layer, it uses the benefit of both convolutional neural network and recurrent neural network models with time complexity of  $O(n)$ , which is straightly associated with the length of the text length.

## 2.10 Recurrent Neural Networks for text classification

**Recurrent Neural Network** (Recurrent NN) is another type of model that shows a time complexity  $O(n)$ . This model investigates a text word to word and saves the semantics of all the past text in a rigid-sized hidden layer [4]. There is no doubt that it has the ability to capture the semantics of a big text but it is a biased model. It means that it focuses on later words than earlier words that cause to minimize the efficiency of capturing the semantics of a whole document as all the words have the same probability to appear in the sequences of words.

In [17],[15] have been proposed various methods and models for classification of text based on recurrent neural network.

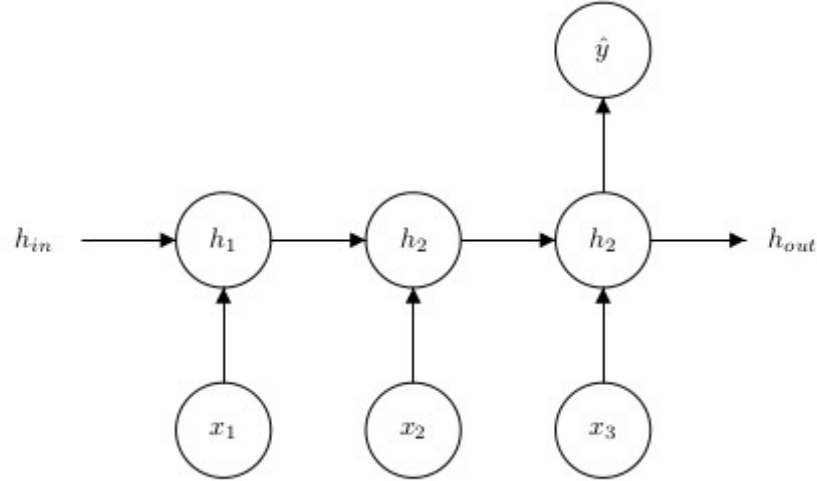


Figure 2.11: Recurrent Neural Network for classification

The activation of the hidden state  $h_t$  is calculated as a function  $f$  of the input  $x_t$  and the previous hidden state  $h_{t-1}$ .

$$h(t) = \begin{cases} 0 & : z = o \\ (f(h_{t-1}, x_t)) & : otherwise \end{cases}$$

Long short-term memory networks (LSTMs) are special kind of networks that allowing information to preserve. The LSTM recurrent neural network equations for classification of texts are following:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1}) \quad (2.50)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1}) \quad (2.51)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_{t-1}) \quad (2.52)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1}) \quad (2.53)$$

$$c_t = i_t * \tilde{c}_t + f_t^i * c_{t-1} \quad (2.54)$$

$$(2.55)$$

where  $i_t$  is input gate,  $f_t$  the activation of memory cells' forget gates,  $o_t$  is output gate,  $\tilde{c}_t$  the states value of memory cell,  $c_t$  the memory cells' new state,  $x_t$  is the input at time  $t$ ,  $W_i$ ,  $U_i$ ,  $V_i$ ,  $W_f$ ,  $U_f$ ,  $V_f$ ,  $W_o$ ,  $U_o$ ,  $V_o$  are weight matrices and  $\sigma$  denotes the logistic sigmoid function and  $*$  denotes element-wise multiplication. The loss function for the classification is trained to minimise the cross-entropy between output and target distributions:

$$L(\hat{y}, y) = - \sum_{i=1}^N \sum_{j=1}^C y_i^j \log \hat{y}_i^j \quad (2.56)$$

where  $\hat{y}_i^j$  is the prediction probabilities,  $y_i^j$  is the desired distribution,  $N$  denotes the number of training samples and  $C$  is the number of class.

# Chapter 3

## Experimental Setup

This chapter provides a description of the data collection and organization, including the division over classes based on semantic reviews. Furthermore, a description is given of the implementation of the Naive Bayes, and of the implementation of the recurrent neural network. Finally, the task is to examine whether a given movie review has a positive or negative sentiment.

### 3.1 Data collection and Data Preparation

The dataset that was used in this thesis contained a collection of 50,000 reviews from Internet Movie Database (IMDB), allowing no more than 30 reviews per movie. The constructed dataset is evenly divided into training and test sets. The dataset contains an even number of positive and negative reviews, 25000 highly polar movie labeled reviews (good or bad) for training, and 25,000 for testing which were manually downloaded from website <sup>1</sup>. A negative review has a score -4 out of 10, and a positive review has a score 4 out of 10. Neutral reviews are not included in the dataset.

---

<sup>1</sup>Dataset and further details are available online at: <http://www.andrew-maas.net/data/sentiment>

Table 3.1: Number of documents per category, classification based on movie reviews

Classification	Class	Documents		Experiment
Movie Review Sentiments	Positive	12,500	training	25,000
	Negative	12,500	testing	25,000
<b>Total</b>		<b>25000</b>		<b>50,000</b>

After all the documents were collected and categorized into two classes, the data was set as input for the algorithms with the removal of all punctuation and special characters along with changing all letters into lowercase form. Traditional stop word removal was not used due to the fact that certain stop words (e.g. negating words) are indicative of sentiment. Stemming was not applied (i.e. removing the ends of words in order to reduce vocabulary size) because the model learns same representations for words of the similar stem when the data suggests it. Furthermore, because certain non-word tokens (i.e. ! and :-)) are suggestive of sentiment, we permit them in our vocabulary. The final form of dataset contained 88,585 total number of unique words. The average review has just under 300 words.

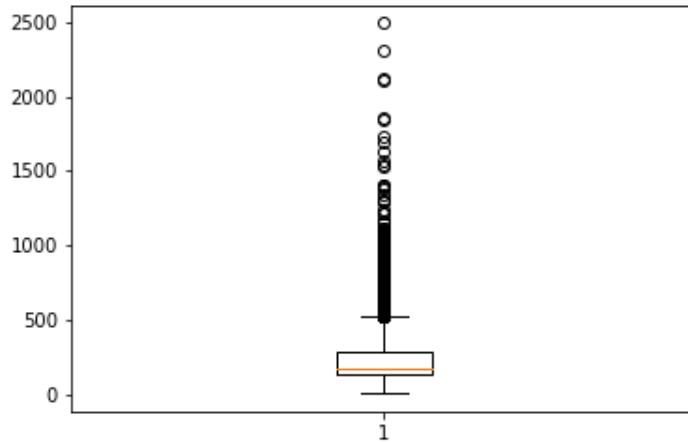


Figure 3.1: Review Length in Words for IMDB Dataset



## 3.2 Naive Bayes

The Naive Bayes estimation requires the proportion of documents in each class in the estimation of the most likely class label. Moreover, for each word the number of occurrences of that word in each labeled class and the total amount of words lies in each class, (see segment (2.8)). During the experiments, for each word observed in the training data, these values (frequencies) were basically enumerated and stored. Since the Naive Bayes classifier is not impervious to zero-events (non-occurrence words), Laplace smoothing was used to increment all counts with a value of 1. In addition, logarithmic scales were utilized to represent probability scores, by preventing probabilities from becoming too small to use for calculations.

It is a type of classifier which is actually a Bayes algorithm which works on very strong independent assumptions for each feature. This is why it gets its naive name. In naive Bayes, grammar doesn't matter because of two main reasons. First of all, it's not an informative feature. Secondly, it is trained on most common words. As mentioned in section 2.5.1, previous research has successfully employed the use of n-grams to alleviate the loss of context, (such as likeness opinion), and to incorporate short expressions that are all in all firmly associated with certain movie reviews. Accordingly, this technique is applied here as well by increasing the vocabulary size with 2- and 3-gram word sequences. Nevertheless, as early trial runs immediately appeared, the naive Bayes classifier has a tendency to perform better when the vocabulary size is modest. Adding 2- and 3-grams incredibly expanded the total number of words in the vocabulary (expanding the number of types from 85,585 to 827,241), and consequently yielded more regrettable outcomes than would be expected in light of earlier research. So as to maintain an appropriately sized vocabulary while simultaneously incorporating useful n-grams, the decision was made to just incorporate bigrams and trigrams with high discriminative values between the classes. So as to discover these words, a strict procedure of feature selection was employed.

Feature selection is the process of choosing a subset of words from the training data that are most crucial for classification [2]. Three generally utilized feature selection methods are removing infrequent words, removing high occurrence words, and removing words that don't impart high common information to the objective classes [7]. In this thesis two of these strategies were employed, the first is the removal of infrequent words. This implies for all unigrams, bigrams, and trigrams, all words or word combinations that happen less than three times were expelled from the vocabulary.

The second method that was employed is the removal of words that share little common information to the objective classes (desired classes). For each word, counted the number of times this word occurred in each of the classes, as well as the total number of times this word occurred in the training data. These values can be utilized to decide how much discriminative information a specific word holds for a specific class. For instance, an exceptionally normal word like "the" appeared on average (depending upon the congregation) a total number of 38,703 times, and was divided relatively equal up to all classes. Due to its even distribution over the classes, the word "the" offered almost no discriminative power and could easily be removed. Keeping in mind the target goal is to choose just the most useful words, all words that did not represent one specific class for a certain minimum proportion of its occurrences were expelled from the vocabulary.

Since the movie review classification task had only 2 different classes, only a limited number of bigrams and trigrams were included, in order to keep the vocabulary size modest. Three different stop words lists were used during the experiment to get the optimal value. Consequently, in light of different trial runs to observe the optimal values using different stop words lists, unigrams were included when they appeared in one specific class for at least 83.29% of the cases.

### 3.3 tf\*idf Sequence Classifier

The Sequence Classifier is intended to apply the tf\*idf metric to any type of input and look for significant sequences, regardless of whether the aim is to classify languages, text categories or music. Although certain different variations of the tf\*idf metric exist, the application displayed here utilizes just three varieties, called ntl1, ntl2 and ntl3. Every one of the three variations utilize the natural term frequency tf (denoted by n), the logarithmic inverse document frequency idf (denoted by t) and are normalized for the length of the document (denoted by l) [16]. The distinction lies in the application of varying sized n-grams ntl1 utilizes just unigrams, ntl2 utilizes just bigrams and ntl3 utilizes just trigrams.

Contrasted with the other classifier, the Sequence Classifier can work with considerably greater vocabularies without suffering diminishing effects on accuracy, so there was no compelling reason to apply any feature selection techniques. Nevertheless, some form of feature selection is automatically incorporated in the calculation of the inverse document frequency, as the most common terms will get a value of zero. Recall from section the formula for calculating idf:

$$idf_{t_i, D} = \log \frac{|D|}{|d_i \subset t_i|} \quad (3.1)$$

where  $|D|$ -the total number of documents(Corpus), and the absolute symbol dft shows the number of the document in which term  $t_i$  occurs. For terms that are so common that they appear in each document (e.g. words like a or the, "and"), the idf value will be the log of 1 (since  $|D|$  and dft are equal), which is zero. Therefore, these words will have no influence on the total score of the tf\*idf metric. It exhibits that if a word occurs less frequently, it will have high IDF frequency.

### 3.4 Multilayer Neural Network

A Multilayer Perceptron or a multi-layer neural network is a function  $f(x) : R^S \rightarrow R^L$ , where  $S$  is the size of input vector  $f(x)$  and  $L$  is the size of the output vector  $f(x)$ .

In matrix notation:

$$f(x) = \varphi(b^{(2)} + W^{(2)}(\phi(b^{(1)} + W^{(1)}x))) \quad (3.2)$$

where  $b^{(1)}$ ,  $b^{(2)}$  are bias vectors,  $W^{(1)}$ ,  $W^{(2)}$  are weight matrices, and  $\varphi$ ,  $\phi$  are activation functions.

The vector  $h(x) = \varphi(b^{(1)} + W^{(1)}x)$  constitutes the hidden layer. The weight  $W^{(1)} \in R^{S \times S_h}$  is the weight matrix connecting the input vector to the hidden layer. Each column  $W^{(1)}$  represents the weights from the input units to the  $i$ -th hidden unit. ReLU activation function is used because it typically yields to faster training (and sometimes also to better local minima). The output vector is then obtained as:

$$output(x) = \phi(b^{(2)} + W^{(2)}h(x)) \quad (3.3)$$

To train an MLP model, all parameters of the model are learned by using Stochastic Gradient Descent with minibatches. The set of parameters to learn is the set  $\theta = W^{(2)}, b^{(2)}, W^{(1)}, b^{(1)}$ . Obtaining the gradients can be achieved through the backpropagation algorithm (a special case of the chain-rule of derivation).

The proposed model is simple multi-layer perceptron model with a single hidden layer. Word Embeddings is fed as input to the neural network. Word embedding is a technique where words are encoded as real-valued vectors in a high-dimensional space. Discrete words are mapped to vectors of continuous numbers in a high-dimensional space. Word Embeddings layer is the first layer in a model. The layer takes arguments that define the mapping. It also includes the maximum number of expected words called the vocabulary size (e.g. the largest integer value that will

be seen as an integer). The layer also allows specifying the dimensionality for each word vector called the output dimension. A 32-dimension vector is used to represent each word. For an experiment, only top 5,000 most frequent words in the dataset are used to set the vocabulary. So the vocabulary size will be 5,000. A movie review is bounded at 500 words, truncating longer reviews and padding shorter reviews with zero value so that they are all the same length for modeling.

Finally, the MLP model is defined by creating a word embedding layer as the first layer. The word vector size to 32 dimensions and the input length to 500. The output of this first layer would be a matrix with the size  $32 \times 500$ . Embedded layers output will be flattened to one dimension then use one dense hidden layer of 250 units with a rectifier activation function. The output layer has one neuron and will use a sigmoid activation to get the output values between 0 and 1 (probability-like values) as predictions. A batch size of 128 is used for training the model. Adam algorithm is used in training because it gives best solutions by controlling the learning rate [11]. Basically, it uses moving averages of the parameters (momentum) that allow it to use a larger effective step size, and the algorithm will converge to this step size without fine tuning. After the model is trained, evaluation is done to check its accuracy on the test dataset. The proposed model achieves a score of 87.16% accuracy. A greater accuracy can be achieved by training this network using a larger embedding and with the addition of more hidden layers.

### **3.5 Convolutional Neural Network**

Convolutional neural networks (CNN), in biological terms, are inspired variants of MLP. They were designed to honour the spatial structure in image data while being vigorous to the position and orientation of learned objects in the picture. This same

idea can be used on sequences, such as the one-dimensional sequence of words in a movie review. This is how the same properties that make the CNN model more useful for learning to identify objects in pictures can assist to learn pattern (structure) in paragraphs of words, namely the methods invariance to the specific position of features.

Word Embeddings is fed as input to the convolutional neural network. Word Embeddings layer is the first layer in a model. The layer takes arguments that define the mapping. It also includes the maximum number of expected words called the vocabulary size (e.g. the largest integer value that will be seen as an integer). The layer also allows specifying the dimensionality for each word vector called the output dimension. A 32-dimension vector is used to represent each word. For an experiment, only top 5,000 most frequent words in the dataset are used to set the vocabulary. So the vocabulary size will be 5,000. A movie review is bounded at 500 words, truncating longer reviews and padding shorter reviews with zero value.

Eventually, define a convolutional neural network model for the experiment. This time, after the Embedding input layer, a Conv1D layer is inserted. This Conv1D layer has 32 feature maps and reads 3 (kernel size) vector elements of the word embedding at a time. The convolutional layer is followed by a 1D max pooling layer with a length and stride of 2 that halves the size of the feature maps from the convolutional layer. The rest of the network is the same as the MLP. It is important to note that Conv1D layer conserves the dimensionality of Embedding input layer of 32-dimensional input of 500 words. The pooling layer compresses this representation by halving it. Running the experiment offers a small but welcome improvement over the neural network model.

One main feature of CNNs is that units share weights, which greatly minimize the amount of computation needed for the model training. It is also seen that CNNs are better at capturing the spatial relationships between words. After training and testing the neural network, the model achieves an accuracy of 87.71%. In future, it is worthwhile to work on the proposed idea. The idea is to set the max pooling layer to use an input length of 500 words. This would compress each feature map to a single 32 length vector and may give even better results

### **3.6 LSTM For Sequence Classification**

Sequence classification is a predictive modeling problem. If a sequence of words is given to the model as input, the task is to predict a category (class) for given the sequence. It is not a trivial case to perform because the issue is that the sequences can vary in length, be comprised of a very large vocabulary of input symbols. Different sequences may have different length of words. When these sequences fed to a neural network model, it may require the model to learn the long-term context or dependencies between patterns (symbols) in the input sequence. LSTM recurrent neural network models are used for sequence classification in a movie review dataset.

A movie review is a variable sequence of words. It means that each movie review contains different amount of words and the sentiment of each movie review must be classified. The words have been substituted by integers that exhibit the ordered frequency (how many times a word occurred) of each word in the dataset. In each review, consequently the sentences are made up of a sequence of integers. For an experiment, only top 5,000 most frequent words in the dataset are used to set the vocabulary. So the vocabulary size will be 5,000.

For an experiment, only top 5,000 most frequent words in the dataset are used to set the vocabulary. So the vocabulary size will be 5,000. A movie review is bounded at 500 words, truncating longer reviews and padding shorter reviews with zero value so that they are all the same length for modeling. Word Embeddings layer is the first layer in a model. It uses 32 length vectors to indicate each word. The next layer is the LSTM layer that contains 100 memory units (called smart neurons). A dense output layer with a single neuron and a sigmoid activation are used because this is a binary classification problem. A sigmoid activation function is used to make 0 or 1 (probability-like values) predictions for the two classes (good and bad). A huge batch size of 64 reviews is utilized to space out weight updates.

In the end, a model is created for an experiment. The model is fit for a small epochs only. The reason is that it quickly overfits the problem. We achieve near state-of-the-art results on the IMDB problem with LSTM performing little tuning. Once fit, an evaluation procedure is performed to estimate the performance of the model on unseen reviews. The model achieves an accuracy of 88.03%.

Recurrent Neural networks (RNN) like LSTM generally have the problem of overfitting. Dropout is a powerful technique for combating overfitting in LSTM models. Dropout can be applied between layers of the neural network. Dropout is applied by adding new Dropout layers between the Embedding and LSTM layers and the LSTM and Dense output layers in different experiments. In first experiment, the model gets the accuracy of 87.17% that indicate a slightly slower trend in convergence compared to the simple LSTM case. Another technique is adapted to compare the results, add dropout to the input and recurrent connections of the memory units with the LSTM precisely and separately. In the second experiment, the model gets the accuracy of



86.44%. It can be seen that LSTM specific dropout has a more pronounced effect on the convergence of the network than the layer-wise dropout.

### 3.7 LSTM and CNN For Sequence Classification

Convolutional neural networks (CNN) proficient at learning the spatial pattern (structure) in input data but LSTM needs to be larger and trained for longer to achieve the same skill. The IMDB review data does have a one-dimensional spatial pattern (structure) in the sequence of words in movie reviews. CNN may be able to select invariant features for positive and negative sentiment. This learned spatial features may then be learned as patterns (sequences) by an LSTM layer. It is easily possible to add a one-dim CNN and max-pooling layers after the Embedding layer. Subsequently, this Embedding layer feeds the consolidated features to the LSTM. A fairly small set consist of 32 features with a small filter length of 3 (kernel size) is used. In the next step, the pooling layer utilized the standard length size of 2 to reduce (halve) the size of the feature map. The rest architecture is same as explained above. The model gets the accuracy of 86.74%. It can be seen that the model achieves similar results to the LSTM for Sequence Classification with Dropout” although with fewer weights and faster training time.

# Chapter 4

## Results

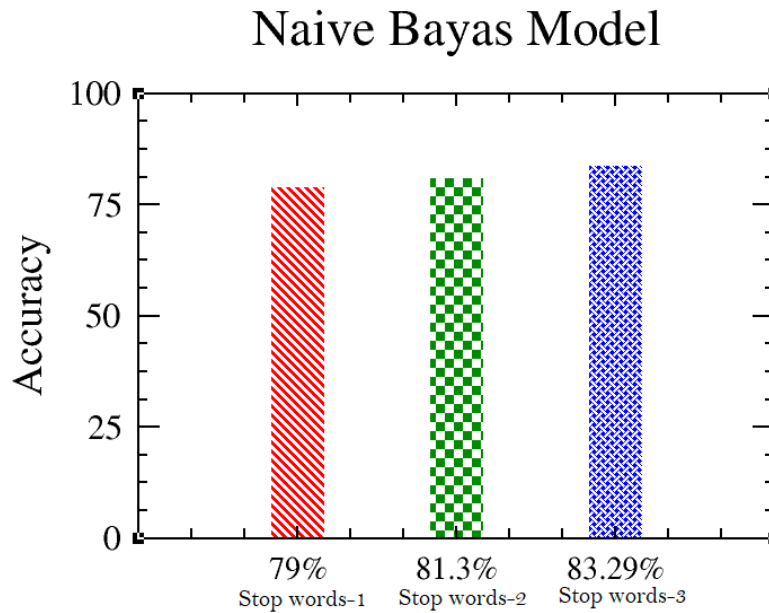
Since the aim of this thesis is to classify texts of IMDB movie review dataset. The results will be judged separately so that the scores of different techniques can be compared. The results are obtained using the commonly applied method of tenfold cross-validation (as mentioned in section 3.1), which calculates average classification accuracy, with accuracy simply defined as the number of correctly predicted labels. For evaluation purposes, accuracy scores are often compared to the majority Baseline, which is the accuracy score obtained when the label from the largest class (i.e. the label with the highest prior probability) is assigned to each document.

The first section of this chapter describes the results achieved by the Nave Bayes, Multilayer Neural Network, Convolutional Neural Network, LSTM for Sequence Classification, LSTM and Convolutional Neural Network For Sequence Classification. The final section concludes with a comparison of all the results and conclusion.

### 4.1 Naive Bayes (multiple entry on text)

In the first experiment, the Nave Bayes was applied to the IMDB data on three different stop words list; once with unigrams, once with combinations of unigrams and

bigrams, and once with combinations of unigrams, bigrams and trigrams as features for classification. The lists of stop words were selected from different sources. For one experiment, they were taken from NLTK(Natural Language Toolkit). For other experiments, we used Google stop words file for the experiment. Finally, We assembled different stop words without order consideration from Google and NLTK files and measure the difference between them in terms of accuracy. The Google and mixed stop words file exhibited not fair results. On the other hand, NLTK stop words file indicated pretty good results for sentiment classification. The graph 4.1 shows the resulting accuracies on the sentiment classification task.



## 4.2 Multilayer Neural Network

Multilayer neural network was applied to the IMDB dataset. The data distribution was in such away that 50% data was consisted on training set and 50% testing set. The model achieves a score of 87.16% accuracy after two epochs. It is a good predicting

model as compared to the traditional machine learning techniques. A fairly good results can also be achieved after several epochs.

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 500, 32)	160000
flatten_1 (Flatten)	(None, 16000)	0
dense_1 (Dense)	(None, 250)	4000250
dense_2 (Dense)	(None, 1)	251
Total params: 4,160,501		
Trainable params: 4,160,501		
Non-trainable params: 0		
Train on 25000 samples, validate on 25000 samples		
Epoch 1/2		
36s - loss: 0.5186 - acc: 0.7060 - val_loss: 0.2986 - val_acc: 0.8713		
Epoch 2/2		
36s - loss: 0.1917 - acc: 0.9266 - val_loss: 0.3098 - val_acc: 0.8716		
Accuracy: 87.16%		

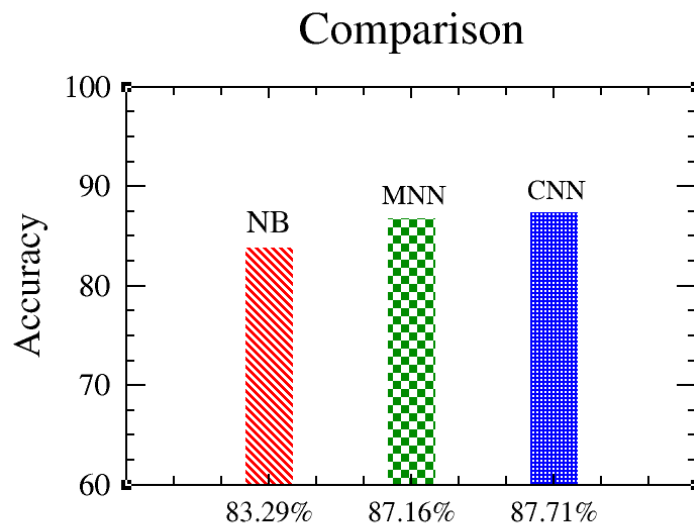
Figure 4.1: Multilayer Neural Network

### 4.3 Convolutional Neural Network

Convolutional neural network was applied to the IMDB dataset. The data distribution was 50% for both training and testing. One-dimensional convolution layer was used along with the maximum pooling layer of size 3. The model achieves an accuracy of 87.71% after two epochs. Convolutional neural network presented better results as compared to the Multilayer Neural Network. It can be concluded that CNN show better performance than MNN after the same number of epochs because it has good architecture to highlight the text patterns (sequences) in the training.

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 500, 32)	160000
conv1d_1 (Conv1D)	(None, 500, 32)	3104
max_pooling1d_1 (MaxPooling1	(None, 250, 32)	0
flatten_2 (Flatten)	(None, 8000)	0
dense_3 (Dense)	(None, 250)	2000250
dense_4 (Dense)	(None, 1)	251
Total params: 2,163,605		
Trainable params: 2,163,605		
Non-trainable params: 0		
Train on 25000 samples, validate on 25000 samples		
Epoch 1/2		
42s - loss: 0.4480 - acc: 0.7630 - val_loss: 0.2936 - val_acc: 0.8774		
Epoch 2/2		
46s - loss: 0.2399 - acc: 0.9054 - val_loss: 0.2912 - val_acc: 0.8771		
Accuracy: 87.71%		

Figure 4.2: Convolutional Neural Network



## 4.4 LSTM For Sequence Classification

Recurrent neural network was applied to the IMDB dataset for sequence classification. The data distribution was 50% for both training and testing. The LSTM layer of 100 memory units (called smart neurons) was used for the experiment. The LSTM model achieves an accuracy of 88.03% after three epochs. Recurrent neural network showed good performance for sequence classification. It can be concluded that LSTM show better performance because its architecture contain memory cells that can memorize and forget text patterns (sequences) from IMDB dataset.

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 500, 32)	160000
lstm_2 (LSTM)	(None, 100)	53200
dense_2 (Dense)	(None, 1)	101

-----  
Total params: 213,301  
Trainable params: 213,301  
Non-trainable params: 0

None  
Train on 25000 samples, validate on 25000 samples  
Epoch 1/3  
25000/25000 [-----] - 734s - loss: 0.5560 - acc: 0.6960 - val\_loss: 0.5335 - val\_acc: 0.7377  
Epoch 2/3  
25000/25000 [-----] - 749s - loss: 0.3263 - acc: 0.8623 - val\_loss: 0.2952 - val\_acc: 0.8807  
Epoch 3/3  
25000/25000 [-----] - 747s - loss: 0.2343 - acc: 0.9080 - val\_loss: 0.2959 - val\_acc: 0.8803  
Accuracy: 88.03%

Figure 4.3: Recurrent Neural Network

LSTM can overfit very quickly. We used dropout layer to avoid over-fitting.

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 500, 32)	160000
lstm_2 (LSTM)	(None, 100)	53200
dense_2 (Dense)	(None, 1)	101

-----  
Total params: 213,301  
Trainable params: 213,301  
Non-trainable params: 0

None  
Epoch 1/3  
25000/25000 [-----] - 631s - loss: 0.5105 - acc: 0.7484  
Epoch 2/3  
25000/25000 [-----] - 687s - loss: 0.3761 - acc: 0.8408  
Epoch 3/3  
25000/25000 [-----] - 681s - loss: 0.3294 - acc: 0.8634  
Accuracy: 86.44%

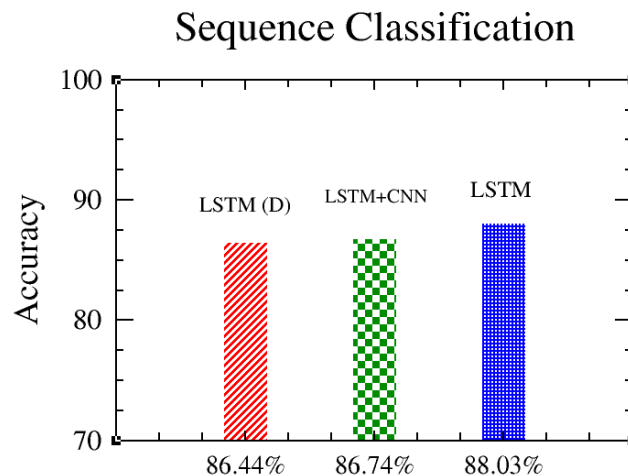
Figure 4.4: LSTM with dropout

## 4.5 LSTM and CNN For Sequence Classification

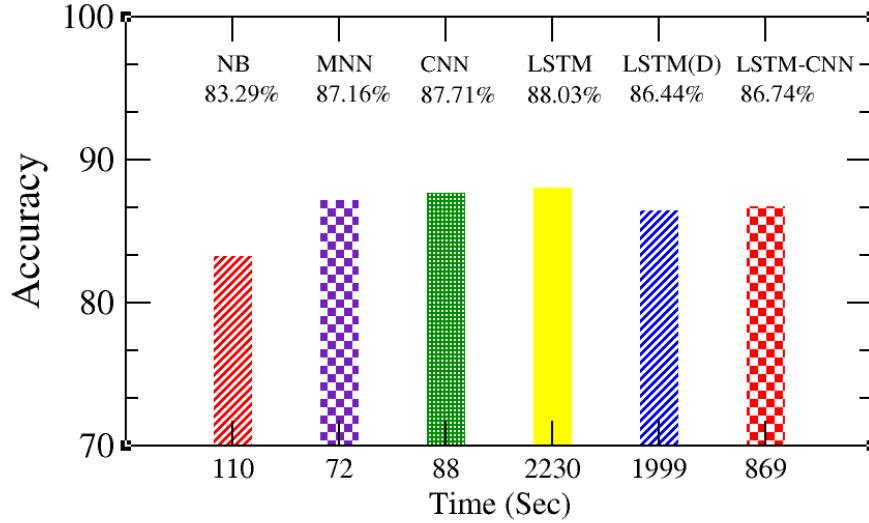
LSTM and Convolutional neural networks were applied to the IMDB dataset for sequence classification. The data distribution was 50% for both training and testing. The model achieves an accuracy of 86.74% after three epochs for sequence classification. It can be seen that LSTM-CNN show not fair performance.

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 500, 32)	160000
conv1d_1 (Conv1D)	(None, 500, 32)	3104
max_pooling1d_1 (MaxPooling1D)	(None, 250, 32)	0
lstm_1 (LSTM)	(None, 100)	53200
dense_1 (Dense)	(None, 1)	101
Total params: 216,405		
Trainable params: 216,405		
Non-trainable params: 0		
None		
Epoch 1/3		
25000/25000 [=====] - 293s - loss: 0.4228 - acc: 0.7910		
Epoch 2/3		
25000/25000 [=====] - 287s - loss: 0.2539 - acc: 0.9003		
Epoch 3/3		
25000/25000 [=====] - 289s - loss: 0.2120 - acc: 0.9194		
Accuracy: 86.74%		

Figure 4.5: LSTM and Convolutional Neural Network



We have used different techniques to perform text categorization task on IMDB dataset. If we compare all the results, we can see that CNN show best performance for word classification. On the other hand, the LSTM model achieves good performance for sequence classification. Subsequently, we show their training time and compare their performances.



## 4.6 Conclusion

The central goal of this thesis was to investigate to what extent a word-based classification approach could successfully predict to what label of a text belongs. For this purpose, four different experiments were executed, each one designed to answer one of the four research questions. We have reached the following conclusions

- We have shown that Naive Bayes classifier could successfully predict sentimental class labels. Based on the results of the first experiment it can be concluded that Naive Bayes is capable of classifying sentimental labels. Furthermore, accuracy results can be improved by providing the classifier with extra features in the form of bigrams and trigrams when predicting sentimental labels. Compared to



the other classifiers, Naive Bayes produces less accurate results when predicting labels.

- The results from the second experiment show that Multilayer neural network and convolution neural network produce better results than the traditional machine learning methods. However, better solutions can be created to achieve better accuracy -we will try to create a better solution in our future work.
- Using LSTM and CNN for sentence classification it was detected that both produce good results to select the correct sentimental class label. However, they can attain even more higher accuracies -we will try to create a better solution in our future work.
- Using all the proposed models lead to above 80% accuracies almost as good as human annotators. That means that our models have good accuracies when classifying sentiment of text.

# Bibliography

- [1] Aggarwal, C. C.; Zhai, C. (2012). A survey of text classification algorithms. In *Mining text data*. Springer. pp 163-222.
- [2] Aved'yan, E. Learning Systems. *Springer*, pp. 88-99.
- [3] Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. (2011). Natural language processing (almost) from scratch. *JMLR* 12:2493-2537.
- [4] Elman, J. L. (1990). Finding structure in time. *Cognitive science* 14(2): pp 179-211.
- [5] Elworthy, D. (1994). Does Baum-Welch re-estimation help taggers? *Proceedings of the Fourth ACL Conference on Applied Natural Language Processing*, pp. 53-58.
- [6] Frank, E.; and Bouckaert, R.R.(.). Naive Bayes for Text Classification with Unbalanced Classes.
- [7] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Machine Learning: ECML-98, Tenth European Conference on Machine Learning*, pp. 137-142.
- [8] Jurafsky, D.; Martin, J.H.(2000). Speech and Language Processing. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition* (Second edition). Upper Saddle River: Prentice Hall.
- [9] Kalchbrenner, N.; Blunsom, P. (2013). Recurrent convolutional neural networks for discourse compositionality. In *Workshop on CVSC*, pp 119-126.
- [10] Kim, Y.; (2014). Convolutional Neural Networks for Sentence Classification. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, pp 1746-1751.
- [11] Kingma, P.D; Ba, L.J.(2015). ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. *ICLR*.

- [12] Kivinen, J.; Warmuth, M.K.; Aue, P. (1995). The perceptron algorithm versus winnow: linear versus logarithmic mistake bounds when few input variables are relevant. In *Proceedings 8th Annual Conference on Computational Learning Theory* (ACM, New York) pp 289-296.
- [13] Lai, S.; Xu, L.; Liu, K.; Zhao, J.; (2015). Recurrent Convolutional Neural Networks for Text Classification. In *Association for the Advancement of Artificial Intelligence*,.
- [14] Lifshitz, Y. (2006). Support Vector Method. Lecture number 7 course. algorithms for the Internet.
- [15] Liu, P.; Qiu, X.; Huang, X. (2016). Recurrent Neural Network for Text Classification with Multi-Task Learning.
- [16] Manning, C.D.; Raghavan, P.; Schütze, H. (2000). An introduction to the Information Retrieval. *Cambridge University Press*.
- [17] Margarit, H.; Subramaniam, R. A Batch-Normalized Recurrent Network for Sentiment Classification.
- [18] McCallum, A.; Nigam, K. (1998). A comparison of event models for naive Bayes text classification. *AAAI-98 workshop on learning for text categorization, Vol. 752*, pp. 41-48.
- [19] Merialdo, B. (1994). Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2), 155-171.
- [20] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In *Proceedings of workshop at ICLR*.
- [21] Pennington, J.; Socher, R.; Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532-1543.
- [22] Ramage, D. (2007). Hidden Markov Models Fundamentals. *Stanford CS299 Section Notes*.
- [23] Seymore, K.; McCallum, A.; Rosenfeld, R. (1999). Learning hidden Markov model structure for information extraction. *Machine Learning for Information Extraction: Papers from the AAAI Workshop*. Tech. rep. WS-99-11, AAAI Press.
- [24] Tan, S. (2006). An effective refinement strategy for KNN text classifier. *Expert Systems with Applications*, 30(2006), pp 2902-298.