SENTIMENT ANALYSIS WITH RECURRENT NEURAL NETWORKS ON TURKISH REVIEWS DOMAIN

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DARKHAN RYSBEK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
SCIENTIFIC COMPUTING

MAY 2019

Approval of the thesis:

SENTIMENT ANALYSIS WITH RECURRENT NEURAL NETWORKS ON TURKISH REVIEWS DOMAIN

submitted by **DARKHAN RYSBEK** in partial fulfillment of the requirements for the degree of **Master of Science in Scientific Computing Department, Middle East Technical University** by,

Prof. Dr. Ömür Uğur
Director, Graduate School of **Applied Mathematics**

Assoc. Prof. Dr. Hamdullah Yücel
Head of Department, **Scientific Computing**

Prof. Dr. Ömür Uğur
Supervisor, **Scientific Computing, METU**

Examining Committee Members:

Prof. Dr. Kasırga Yıldırak
Department of the Actuarial Sciences, Hacettepe University

Prof. Dr. Ömür Uğur
Institute of Applied Mathematics, METU

Assoc. Prof. Dr. Ümit Aksoy
Department of Mathematics, Atılım University

Assoc. Prof. Dr. Hamdullah Yücel
Institute of Applied Mathematics, METU

Assist. Prof. Dr. Büşra Zeynep Temoçin
Institute of Applied Mathematics, METU

Date:

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

Name, Last Name: DARKHAN RYSBEK

Signature :

ABSTRACT

SENTIMENT ANALYSIS WITH RECURRENT NEURAL NETWORKS ON TURKISH REVIEWS DOMAIN

RYSBEK, DARKHAN
M.S., Department of Scientific Computing
Supervisor : Prof. Dr. Ömür Uğur

May 2019, 57 pages

Easier access to computers, mobile devices, and availability of the Internet have given people the opportunity to use social media more frequently and with more convenience. Social media comes in many forms, including blogs, forums, business networks, review sites, and social networks. Therefore, social media generates massive sources of information in the shape of users‘ views, opinions, and arguments about various products, services, social events, and politics. By well-structuring and analysing this kind of data we can obtain significant feedbacks about products and services. This area of research is typically called sentiment analysis or opinion mining. In the last decade, this field of Natural Language Processing (NLP) has witnessed a fascinating progress due to Deep Neural Networks (DNNs).

Recurrent Neural Networks (RNNs) are one of the main types of DNN architectures which are used at modelling units in sequence. They have been successfully used for sequence labelling and sequence prediction tasks, such as handwriting recognition, language modelling, machine translation, and sentiment analysis.

Most of the studies carried on sentiment analysis using RNNs have been focused on English texts and some researches have studied on different languages. In this thesis, sentiment classification using RNNs is applied on Turkish reviews domain. Additionally, different types of word representations are used to achieve acceptable results. This dissertation presents a description of the considered model architectures and comparison of them with various word representations on two Turkish movie reviews datasets. Generally, our experimental results show that RNN models achieve reasonably good results on Turkish texts as on English texts and choice of different word representations can improve the performance of the approaches.

Keywords: Sentiment Analysis, Natural Language Processing, Deep Neural Networks, Recurrent Neural Networks, Machine Learning, Turkish

ÖZ

TÜRKÇE YORUMLAR ALANI ÜZERİNDE ÖZYİNELİ SINIR AĞLARI İLE DUYGU ANALİZİ

RYSBEK, DARKHAN
Yüksek Lisans, Bilimsel Hesaplama Bölümü
Tez Yöneticisi : Prof. Dr. Ömür Uğur

Mayıs 2019, 57 sayfa

İnternetin kullanılabilirliği ve bilgisayarlara, mobil cihazlara daha kolay erişim insanlara sosyal medyayı daha sık ve daha rahat kullanabilme fırsatı verdi. Sosyal medya, bloglar, forumlar, iş ağları, eleştiri siteleri ve sosyal ağlar dahil olmak üzere pek çok biçimde gelir. Bu nedenle, sosyal medya çeşitli ürün, hizmet, sosyal olaylar ve politika hakkında kullanıcıların görüşleri, düşünceleri ve tartışmaları şeklinde muazzam bir bilgi kaynağı oluşturur. Bu tür verileri iyi yapılandırarak ve analiz ederek, ürünler ve hizmetler hakkında önemli geri bildirimler elde edebiliriz. Bu araştırma alanı genellikle duygu analizi veya düşünce madenciliği olarak adlandırılmaktadır. Derin Sınır Ağları (DNNs) nedeniyle bu Doğal Dil İşleme alanı son on yılda büyüleyici bir ilerlemeye tanık olmuştur.

Özyineli Sınır Ağları (RNNs), modelleme birimlerinde dizi ile kullanılan DNN mimarileri türlerinden biridir. El yazısı tanıma, dil modelleme, makine çevirisi ve duygu analizi gibi dizi etiketlendirme ve dizi öngörü görevleri için başarıyla kullanılmıştır.

RNN'leri kullanarak duygu analizi üzerinde yapılan çalışmaların çoğu İngilizce metinlere odaklanmıştır. Birkaç araştırmalar değişik dillerde yapıldı. Bu tez kapsamında, RNN'leri kullanılarak duygu sınıflandırması Türkçe yorum alanına uygulanmıştır. Ek olarak, kabul edilebilir sonuçlar elde etmek için farklı türlerde kelime gösterimleri kullanılmıştır. Bu tez, ele alınan model mimarilerinin bir tanımını ve bunların iki Türk film incelemesi veri setinde çeşitli kelime temsilleriyle karşılaştırılmasını sunmaktadır. Genel olarak, deneysel sonuçlarımız RNN modellerinin, Türkçe metinlerinde İngilizce metinlerde olduğu gibi oldukça iyi sonuçlar elde ettiğini ve farklı kelime gösterimlerini seçmenin yaklaşımların performansını artırabileceğini göstermektedir.

Anahtar Kelimeler: Duygu Analizi, Doğal Dil İşleme, Derin Sınır Ağları, Özyineli Sınır Ağları, Makine Öğrenme, Türkçe

To my family

ACKNOWLEDGMENTS

I would first like to my supervisor Prof. Dr. Ömür Uğur who has provided me with invaluable guidance and persistent support during my research.

I would also like to thank Prof. Dr. Ismail Hakkı Toroslu for his help and ideas. It has been very good to have the opportunity to get advice if any problems were encountered with either the data or the machine learning procedure.

I want to thank my friends and colleagues for their advices and encouragements throughout my academic career.

My appreciation also extends to my sister who supported me morally during my education.

Above all, I would like to thank my parents for all the unconditional support throughout all these years, and also for giving me the opportunity to study in this university.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AdaGrad	Adaptive gradient algorithm
Adam	Adaptive Moment Estimation
ANN	Artificial Neural Network
ASCII	American Standard Code for Information Interchange
BDLSTM	Bidirectional Long Short-Term Memory
BRNN	Bidirectional Recurrent Neural Network
BPTT	Backpropagation Through Time
BoW	Bag-of-Words
CBOW	Continuous Bag-of-Words
CE	Cross Entropy
CNN	Convolutional Neural Network
CNTK	Microsoft Cognitive Toolkit
DL	Deep Learning
GRU	Gated Recurrent Units
LSTM	Long Short-Term Memory
ME	Maximum Entropy
MSE	Mean-Squared Error
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Network
NLP	Natural Language Processing
POS	Part of Speech
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SO	Semantic Orientation
SVM	Support Vector Machines
URL	Uniform Resource Locator
VSM	Vector Space Models

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

In human life the role of language is very significant because it is an essential tool for people to communicate with each other in our society. Human being has always been absorbed with supernatural ideas, one of which was the creation of intelligent robots. To make a machine to be considered intelligent, the well-known Turing test stated that machines must have capacities as much as human [43]. That is how the interest of embedding human language into machines, Natural Language Processing (NLP), the one of the study area of Artificial Intelligence came out.

Machine translation was the first field of natural language processing where research studies initially started with the translation of 60 Russian sentences by the Georgetown - IBM experiment in 1954 [34]. However, researchers acknowledged that the existing linguistic system was not enough to be successful and more advanced methods to understand linguistic unit were needed. After a quarter century later with the evolution of machine learning, researchers made sufficient breakthrough in the areas of NLP. Machine learning approaches become suitable tools for computers to learn and incorporate the rules themselves. The final step of NLP to be successful came with by the development of the Internet and its infinite supply of data resources.

With the advent of social media resources on the Internet, which include social networks, forums, blogs, and so on, there has been a raise in the number of users of these resources. Meanwhile, people actively share their opinions and views in comments, reviews, and discussion forums with their phones, laptops, and tablets. Therefore, the amount of data collected from social media has been unbelievably increased. For example, according to reliable resources each day 500 million Tweets and 4.3 billion Facebook messages are shared [77]. So, millions of positive, negative or neutral comments are added every day. By well-structuring and analysing this kind of data we can obtain significant feedbacks about products and services. This area of research is typically called sentiment analysis or opinion mining. Interest in this area has been actively manifested in recent years and has been applied in various domains by using different methods. There are several annual competitions. For example, the competition to create automatic systems for sentiment analysis in English called SemEval, which has been taking place since 2010.

At the first time it seems that machine learning approaches were the best choice for sentiment analysis but later researches have been using state-of-the-art deep learning models which have been showed excellent results [41, 18, 80]. There are a lot of researches for sentiment analysis applied to English texts whereas for Turkish texts there have been only a few studies [20, 40, 44]. In this study, the problem of classifying Turkish texts by sentiments will be considered. For classification part we use deep learning models such as LSTM and bidirectional LSTM. Earlier, models based on RNN were not used so much for the sentiment classification of the Turkish texts, although they managed to show impressive results in the tasks of classifying English texts. The goal of this thesis is to find the performance of these models on Turkish reviews and to compare the results with the other machine learning technique based on the same dataset.

1.2 Sentiment Analysis

Term *Sentiment* is mainly related with the words as feeling, emotion or opinion which are subjective responses.

Sentiment analysis is a subfield of natural language processing that focuses on determine the polarity, sentiment or emotion of events and, through the detected word features assign them into, for instance, three category of polarity as positive, negative or neutral. Generally in practice, the semantic or polarity of a given text chooses a binary categorization, instead of using three categories, as:

- positive or negative;
- like or dislike;
- good or bad.

Furthermore, there exists various levels of sentiment analysis; most popular of them are *document level* sentiment analysis, *sentence level* sentiment analysis, and *aspect level* sentiment analysis [51]. Studies [63] and [85] are based on document level sentiment analysis where they discuss the polarity of whole document. Pang and Lee in their study [62] explain different applications of sentiment analysis in various domains and applications:

- to review-related websites;
- as a sub-component technology like recommendation systems, flame detection, question answering systems, summarization, citation analysis;
- in business and government intelligence;
- across different domains like politics, sociology, etc.

The motivation of this thesis is to apply sentiment analysis in review-related websites by using Turkish reviews domain crawled from the well-known Turkish movie website of BeyazPerde [1].

1.3 Structure of the Thesis

The rest of the thesis is organised as follows:

Chapter 2 - Literature Survey provides fundamental concepts of sentiment analysis and its application for Turkish language.

Chapter 3 - Background covers information about different methods and models used in this thesis.

Chapter 4 - Experimental Setup is the part where the dataset, evaluation metrics and frameworks used in experiments are described briefly.

Chapter 5 - Results and Discussions reveals the results of the experiments and discusses these results.

Chapter 6 - Conclusion and Future Work gives a summary and concludes the thesis. Moreover, future work and ideas to improve achievements gained in this thesis are suggested in this chapter.

CHAPTER 2

LITERATURE SURVEY

2.1 Sentiment Analysis and Opinion Mining

This section will introduce the general understanding and principles in the field of sentiment analysis. First of all, the basic definition of sentiment analysis will be described. Then, the concepts of sentiment analysis will be provided. Finally, the types of sentiment analysis which appears in studies of sentiment analysis in different levels: document level, sentence level, and aspect level sentiment analysis.

2.1.1 Definition

Sentiment analysis, also familiar as opinion mining, is the field of study that analyses people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes [52]. Initially, sentiment analysis has been about opinion polarity, i.e., whether someone has positive, neutral, or negative opinion towards something [17].

Sentiment analysis is accepted as a classification problem in the field of computational linguists and sometimes called as sentiment classification. It aims to determine whether a given document or text as a whole express a positive or negative opinion [51]. Sentiment classification has several important characteristics, including various tasks, features, techniques, and application domains [3].

In the past decades, there has been done numerous research works related to sentiment analysis due to the existence of various domains with huge amounts of literature data on document, sentence, phrase, and aspect level analyses. Therefore, it has gained extensive attention and the number of its application domains has increased by the time. Moreover, it has become one of the highly active challenging research areas of NLP in recent years due to a wide variety of practical applications [50].

2.1.2 Concepts in Sentiment Analysis

Generally, sentiment classification problem can be divided into two independent classification problems where the first problem is the subjectivity classification and the second problem is based on the polarity of the opinion. In other words, one is to figure out if a given document or text are subjective or objective and the other is to classify the subjective document into separate categories like positive or negative.

The determination of the distinction between subjective and objective documents is the initial task that has to be considered. In computation linguistics the term subjectivity has different meanings and is usually closely related to the point of view. Lei and Liu in their research work [47] mentioned that due to various facts opinions and sentiments can be involved in objective texts, whereas they cannot be signified in subjective texts. In addition, an automatic processing of objective and subjective documents are occasionally complicated since the documents are mostly not explicitly formulated. Moreover, in some cases the entire document cannot be either objective or subjective text, which demonstrates a new challenge in the form of aspect level of subjectivity analysis.

Despite the fact that the analysis of subjectivity is a complicated work, it is used as a criterion to denote the importance of an expression regarding the sentiment analysis. Therefore, subjectivity analysis has been used as a pre-processing step to clean the data for sentiment classification [49].

On the whole, the determination of polarity is the major area in sentiment analysis. The main task is to identify the polarity of the text and the most commonly used method is the binary orientation, positive or negative, of a subjective approval without regarding the external context of the proposal or document. Furthermore, there exist a few types of scales as continuous [6] and discrete [83] which are used in sentiment analysis studies.

In general, polarity is ranked as a number in [-1,1]: (-1) means *the most negative* polarity and 1 corresponds to *the most positive* polarity. In theory, the center of this interval has the *neutral* polarity however in practice this category is refused in most cases in order to make the problem easier and spelt up into positive and negative categories. In this research, the problem is conducted as a binary classification task which means we identify the polarities in two categories: positive or negative.

2.1.3 Types of Sentiment Analysis

As mentioned earlier, sentiment analysis evaluates the text or document and categorizes it into various classes as positive, negative, and neutral according to opinions expressed in text. Sentiment analysis has been investigated on multiple linguistic levels and most frequently at three levels: document level, sentence level, and aspect level. The basic descriptions are given in Table 2.1.

Table 2.1: Levels of Sentiment Analysis

LEVELS	**DESCRIPTION**
Document level	Classifying the whole document as positive, negative or neutral
Sentence level	Associated with a phrase or sentence
Aspect level	Sentiment on entities and / or aspect of those entities

Document Level Sentiment Classification: The main challenge of document-level sentiment analysis is to determine overall polarity expressed in a whole document. For example, this system would be able to label the overall sentiment polarity of a customer review about a certain product. This level of sentiment analysis assumes that the documents express sentiment towards a single entity such as reviews of products, movies, and hotels. The results of document level sentiment analysis usually have two, positive or negative, or three, positive, negative or neutral, outputs and the average length of a document depends on the domains. However, in some cases like news domains there can be several opinions in one document and criticizes multiple targets.

There are a few number research studies for document level sentiment analysis and various methods to address the problem and improve the accuracy. Supervised and unsupervised learning are the main approaches for document level sentiment analysis. In supervised learning, there are finite set of category outputs for each document and training data accessible for each category. In unsupervised learning, the main idea is to find the Semantic Orientation (SO) of specific phrases within the document, compute the average of SO of these phrases and compare it with some predefined threshold which can then help to label the documents as positive or negative [71].

Sentence Level Sentiment Classification: The sentence level sentiment classification is a fine-grained level than document level sentiment analysis and each sentence has expressed a neutral, positive, or negative opinion. Due to the fact that the sentences are a type of short documents, there is no particular variation between the two described levels of sentiment analysis [51]. The essential part in this level is subjectivity classification in which objective sentences signifying factual information are split up from subjective sentences with sentiment. In addition, each specific sentence is syntactically and semantically connected with other parts of the text. Thus, this level task is desired both local and global contextual information. On the other hand, different strategies are applied to resolve the task with various sentences which can be conditional, question, and even complicated sarcastic sentences [36].

Aspect Level Sentiment Analysis: Unlike an earlier described two levels, aspect level sentiment analysis explores what the holder feels, likes or hates, about the target. To make a prediction it primarily collects information related to a specific entity which generally has many aspects, attributes with different opinions. It appears in discussion forums or in product review blogs where reviews are about products such as phones, cameras, even drugs and so on [73].

At the document level, if the text is positive or negative this does not mean that the entire document is respectively positive or negative because of the existence of aspects with different opinions. Besides the sentence level sentiment analysis often related to subjectivity classification. Therefore, aspect level which performs a more fine-grained analysis than two other levels are explored and has three steps of evaluation: extracting features of target, determining feature-wise polarity, summarizing the overall evaluation. Aspect level sentiment analysis, also called *feature-based* or *entity-based* sentiment analysis, is more challenging task compared to other level of analysis due to it identifies fine-grained opinion polarity towards a specific aspect associated with a given target. Moreover, feature-based sentiment analysis requires more complicated machine learning approaches since basic algorithms cannot deal with complex sentences unfortunately.

Besides, different from the levels described above there are other levels of sentiment analysis. A few number of researches have been made on phrase level [88], clause level [89], and word level [90].

2.2 Sentiment Analysis in Turkish

2.2.1 Turkish Sentiment Analysis

There are many languages in the world and each language has its own grammar rules. Turkish language is one of the grammatically rich languages and morphologically agglutinative language. Its specific characteristics make sentiment analysis problems more complicated to resolve for this language. In order to achieve more excellent results and get more precise sentiment classifications, a powerful sentiment analysis structure specific to Turkish should be created which must handle with different linguistic markers such as negations, conditional constructions and so on. Unfortunately, already demonstrated natural language processing systems for English language cannot directly be applied and then translated into Turkish language because of the several differences between these languages. Some of the main differences are described as follows:

1. *Turkish Alphabet*. As other Turkic languages, Turkish alphabet has several letters which do not exist in English alphabet: ‘ı’,’ğ’,’ü’,’ş’,’ö’,’ç’. In some practical cases users substitute these Turkish characters for their closest ASCII characters, for example, letter ‘ğ’ into ‘g’, and use the new strings which may create a new and more complicated problem, set of meaningless words. There-

fore, a pre-processing step, which is called *deasciification*, is needed that converts Turkish text written with ASCII characters into proper Turkish text with its specific accented letters.

2. *Agglutinative Morphology*. Different from English language in Turkish language, the words can be generated by adding suffixes to a root word instead of adding new several words to the main word. These suffixes can change the part-of-speech tagging or semantic orientation of the word. For example, the word "göz" means "eye" but with the suffix "lük" it changes into "gözlük" which means "glasses". For all agglutinative languages, not just for Turkish language, there are extra additional challenges such that a creation of a reasonable corpus of polarity lexicon which would contain all necessity words of a language.

3. *Negation*. In Turkish language, there are several approaches where negation markers are switched the sentiment polarity of a word: with the affixes "ma/me", "siz/sız" or with the help of another word such as "değil" and "yok". The Table 2.2 shows the words with their meaning in English and some sentences as an example. In the last example, polarity of all sentence switches into negative with the word "başarısız" and then changes to positive with the word "olmadı" [22].

Table 2.2: Samples of Turkish words and sentences

TURKISH	**ENGLISH**
'Bitmedi', 'bağımsız'	'it is not finished', 'independent'
'akıllı değil','parası yok'	'not smart', 'has no money'
Bu filmi izlediğim için pişman değilim.	I don't regret watching this movie.
Sınavdan başarısız olan kimse olmadı.	No one failed the exam.

In most cases, the text data collected from any domains through the Internet is more noisy and has insignificant information. In order to change it to more well-organized data with less noise, there is a need of the pre-processing step which is called data pre-processing. This step usually improves the quality of the dataset and generates more meaningful information from the dataset. In general, the main steps of the data pre-processing step are: tokenization, removal of stop-words, and stemming.

After the pre-processing step is the step of feature extraction, a creation of vector space and feature selection. This step will be described in next chapter.

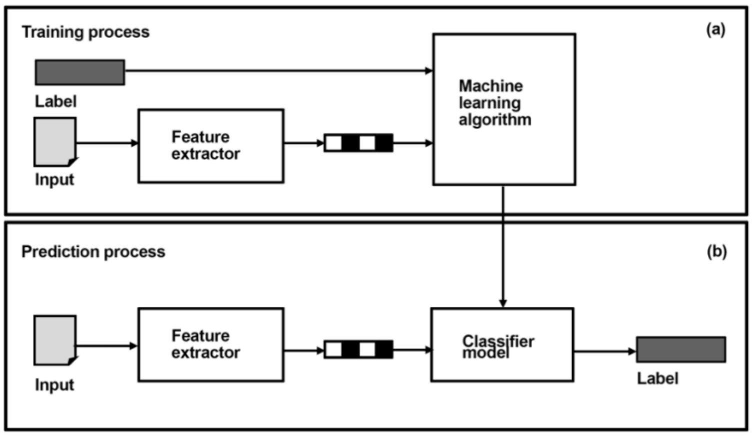


Figure 2.1: A standard sentiment analysis pipeline [38].

Finally, after the feature selection, the training text data would be learned and classified into polarity categories by using the classifier and when the test data arrives it tries to predict correctly as shown in Figure 2.1. Two main categories of sentiment analysis approaches are commonly used as classifiers, which are divided based on the information they use. They are lexicon-based and corpus-based approaches. The former approach essentially computes the polarity for a document by aggregating the semantic orientation of the words, whereas the latter approach uses supervised learning algorithms, which include machine learning or deep learning algorithms, to train a sentiment classifier through the training data. Sometimes hybrid approaches are used which are developed by combining the two approaches [21]. This study relates to corpus-based approach since only deep learning models are used.

2.2.2 The Studies on Turkish Sentiment Analysis

There are a lot of researches done for English language in the fields of natural language processing, especially related to sentiment analysis: not only research works in literature but also necessary resources are established for English language [84, 18, 80]. Research interests on sentiment analysis for non-English languages have increased in recent years. One of them is the Turkish language but there is still not much research in this field.

One of the first studies about sentiment analysis in Turkish is Eroğul‘s master thesis [20]. In this study, as features used n-grams and the effects of part of speech tagging, spellchecking, and stemming are explored by using the Turkish morphological analyser called Zemberek [4]. Moreover, as a dataset for this work used Turkish movie reviews which are crawled from the familiar website Beyazperde [1] and classified by using Support Vector Machines (SVM) at the document level. As a result, 85% of accuracy is achieved on the binary sentiment classification.

Kaya et al. (2012) have investigated sentiment analysis of Turkish political news in online media by using various supervised machine learning algorithms which are Naive Bayes, Maximum Entropy (ME), SVM, and the character based n-gram language models [40]. As classification features frequency of polar word unigrams, bigrams, root words, adjectives, and effective (polar) words are used. In conclusion, they have reported a classification accuracy of 76%-77% with different features and conclude that the maximum entropy and the n-gram language models in comparison with SVM and Naive Bayes methods are more efficient in classifying Turkish political news.

Vural et al. (2013), present a framework for unsupervised sentiment analysis in Turkish text documents [87]. They customized SentiStrength [84], a lexicon-based sentiment analysis library for English that assigns a positive or a negative score to a given text, by translating its polarity lexicon to Turkish. Authors classified their unsupervised framework on the same dataset that was already used in [20] and report an accuracy of 76% on classifying Turkish movie reviews as positive or negative.

Apart from the binary sentiment classification of texts in Turkish, the different areas of the research are also on analysis of emotions. The master thesis of Boynukalın [9] is one of the research related to emotion analysis. She has presented an emotion classification on Turkish texts by using a new dataset and applied machine learning techniques to compare them with each other. To improve the performance she has added new features appropriate with the morphological characteristics of Turkish language.

In spite of the existing works on sentiment analysis of the Turkish language including different levels such as sentence, aspect, and document levels, there is still almost no research by applying the state-of-the-art neural networks. In this study, a well-known recurrent neural networks and its types will be applied to Turkish reviews at document level since they have proved to have better performance for English texts [31].

2.2.3 Natural Language Processing Tool (Zemberek)

There exist various kinds of natural language processing tool for English language. However, for Turkish language there are a few and the most utilized open source tool is Zemberek [4]. It has several capabilities such as morphological analysis, stemming, spellchecking, and part of speech tagging.

One of the most important tools in natural language processing is the stemming process where a morphological parser of Zemberek reduces inflected words to their base or to root form which would be used as features in the experiments.

Another important tool in NLP is the spellchecking step in which the incorrectly spelled words in a text would be checked and replaced with its correct form by using Zemberek library. A spellchecking algorithm under Zemberek allows to process up to 3 inappropriate or incorrect characters in the roots and 2 characters in suffixes. However it has some limitations or drawbacks: it does not fix dates, times and numbers; there may be a lot of word suggestions, and so on. For example, for the word "hoşlanmak" which translates to English as "like", suggestions of the Zemberek library are shown in Table 2.3.

Table 2.3: The suggestions of the Zemberek library for the word "hoşlanmak"

'hoşlaşmak'	'hoşlatmak'
'hoşlamak'	'hoşlansak'
'hoşlanma'	'hoşlanman'
'hoşlanmam'	'hoşlanmaz'
'hoşlanmak'	'boşlanmak'
'haşlanmak'	'hoplanmak'

Part of speech tagging, also familiar as grammatical tagging, is very significant preprocessing task for natural language processing activities that reads text in some language and assigns parts of speech to each word such as noun, verb, adjective, etc. In fact, part of speech tagging process is not just mapping words to their part of speech tags. It is much more complex procedure due to the different meanings of the word in different contexts or domains.

CHAPTER 3

BACKGROUND

3.1 Text Representation

In natural language processing field, the most essential part is the so called *word representation*, the representation of words, sentences or documents in a numerical way since computers cannot recognize and analyse raw text data. When machine learning or deep learning approaches are applied for sentiment analysis tasks, as the input features to represent text data in the model would be taken word representations or word embeddings. There are several types of them: One-Hot Representation, Bag-of-Words, Word2Vec, FastText [8], Glove [67], and others.

3.1.1 One-Hot Representation

The most trivial word representation is the *One-Hot Representation*, which is generally used before neural networks were applied to natural language processing problems. In this method, each word is represented as a vector with the length equal to the size of the vocabulary, which is created by all words after pre-processing in the corpus. The representation vector is consisted of one in the position of the word corresponding to its index in the vocabulary and multiple zeros in the rest of the positions. For example, the vector representation of words “Ankara”, “Turkey” would be as $[0, 0, 0, 1, 0, \ldots, 0, 0]$, $[0, \ldots, 0, 1, 0, \ldots, 0]$ respectively. However, this type of word representation has many drawbacks such as the expensive computation of word similarity and the large size of the vocabulary. Despite the disadvantages, this word representation method can solve some natural language processing problems by using machine learning algorithms like Naïve Bayes [57] and Support Vector Machines (SVM) [37].

In this research, the type of distributed representation, which is known as Word2Vec, is used since the state-of-the-art recurrent neural network approaches are applied for sentiment analysis problem. More about Word2Vec method will be described in the following subsection.

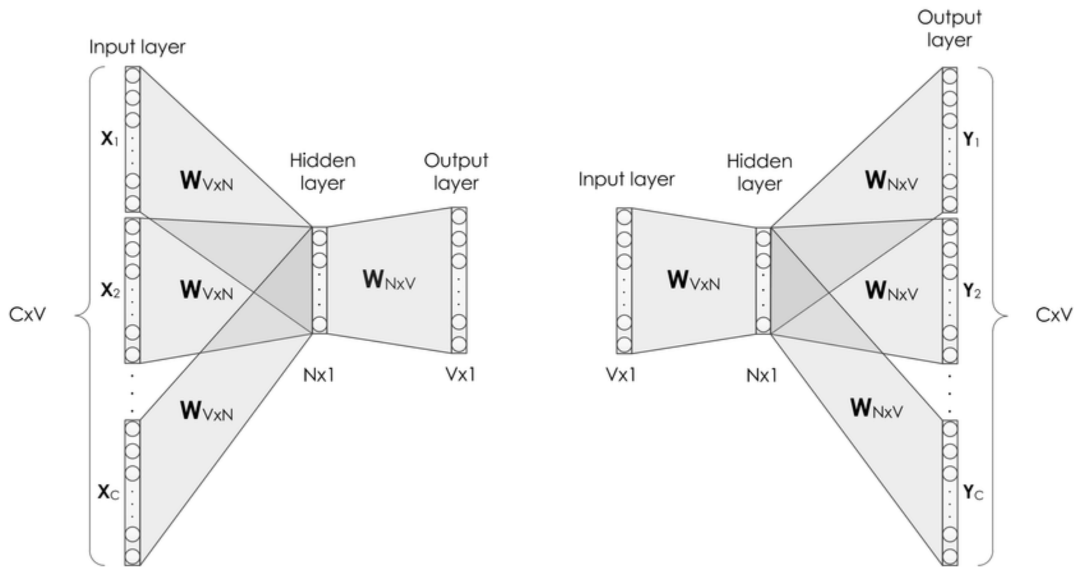
3.1.2 Distributed Representation

Different from one-hot representation and more superior alternative is the distributed representation which was originally introduced by Hinton in 1986 [32] and has been improving its relevance to natural language processing applications in recent years. The distributed representation of words is more familiar as word embeddings and the main purpose is embedding the words into real-valued, dense, and low-dimensional vectors by the meanings of words. As described earlier, the drawback of the one-hot representation is the independence between words in representation space and inability to calculate word similarities. In contrast, word embeddings are allowed to compute similarity of words through low-dimensional matrix operations. Metric used is the cosine similarity due to fact that the semantic similarity between two words is correlated with the cosine of the angle between their word embeddings [48].

In general, the word embeddings are derived by Vector Space Models (VSM) [76]. VSMs are divided into two categories: count-based and predictive models. The former is about the computation of the statistics of co-occurrence of words with its neighbouring words in large text corpus and mapping those statistics into a low dimensional, dense vector. The latter, as its name suggests, tries to predict the word from its neighbours by the learned low dimensional, dense vectors.

One of the most popular models of VSM predictive models is the well-known Word2Vec algorithm which was developed by Mikolov et al. in 2013 [58]. The fundamental idea behind Word2Vec is to predict the surrounding words of each word in a window with the fixed length, instead of capturing directly all co-occurrence counts.

There are two possible variations of learning algorithms to produce a distributed



(a) CBOW (b) Skip-gram

Figure 3.1: CBOW and Skip-gram model architectures [86].

representation of words: Continuous Bag-of-Words model (CBOW) and continuous Skip-gram model. In CBOW architecture (see the Figure 3.1a), the network aims to predict the current word based on the window of n neighbours that occur before and n neighbours that occur after the current word which has been given as an input. In contrast, Skip-gram architecture (see the Figure 3.1b) is the opposite of CBOW model in that, it tries to predict the surrounding $2n$ words with the central target word as input. Finally, the authors of [58] conclude that both models get almost similar results and the difference is mostly depends on the size of the dataset, Skip-gram tends to be a useful model for larger datasets, while CBOW performs slightly better on small datasets.

3.2 Deep Learning

For a long time applying neural network architectures, in comparison with other machine learning models, to solve some problems unbeneficial due to the lack of computational resources and the data. However, it can be seen that the technology has been progressing exponentially with the years that is also affected by the growing of computational infrastructure and by the availability of large amount of good quality of training data. Consequently, the complex neural network models have been exceeded any other machine learning models in various tasks such as image processing, object detection, recommender systems, and others.

In recent years, NNs have been the most widely used approaches for natural language processing tasks such as sentiment analysis, text summarization [74], named entity recognition [45] and question answering [25]. The most popular types of neural networks are Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs).

Next, we provide the necessary background for neural networks and the types of recurrent neural networks.

3.2.1 Neural Networks

Artificial Neural Networks (ANNs), also familiar as Neural Networks (NNs) are machine learning models which are inspired from the human brain. The first models of neural network developed by a neurophysiologist Warren McCulloch and a mathematician Walter Pitts in 1943. In their work [68], they stated how neurons might work. Neural networks consists of the layer of input neurons or signals which can be different feature values, an output layer where the result of the network obtained, and an amount of various hidden layers between the input and output layers. In addition, each layer has a few or several neurons.

Input signals are passed through the network, layer by layer, by using the weighted connections to eventually reach the output layer. At some neurons, a nonlinear function can be triggered. The goal of a learning process is finding weights that would made the neural network demonstrate desired behaviour. This is the example of Multilayer Perceptron (MLP) which is also called Feedforward Neural Networks. A general architecture of artificial neural networks is shown in Figure 3.2.

Despite the fact that the feedforward neural networks have been successfully applied and got better performance in many tasks, it does not consider the transient prospect that characterizes sequential data. That is they are not reasonably good when it comes to the data that depend on the previous data. To solve such kind of problems the neural networks have evolved to so called recurrent neural networks. In the next subsections

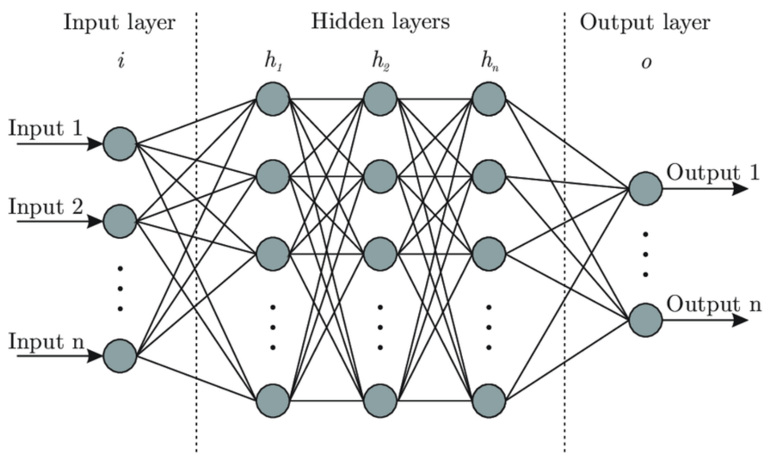


Figure 3.2: Artificial Neural Network with three hidden layers [11].

not only the training process of neural networks would be described briefly in detail, but also the basics of recurrent neural network architectures with its types will be introduced.

3.2.2 Perceptron

The fundamental unit of neural networks is the one of the types of artificial neuron, called perceptron. A perceptron takes a fixed number of inputs and produces a single output. The way of computing the output essentially has several steps such as taking an input, calculating the weighted sum by introducing weights, bias term, and applying activation function. The process described above can be formalized as follows:

The perceptron has n inputs represented as an input vector $x = (x_1, x_2, \ldots, x_n)$. Each input has an assigned weight that defined by a vector of weights $w = (w_1, w_2, \ldots, w_n)$. Consequently, the weighted input values are combined which gives the weighted sum:

$$\varepsilon = w \cdot x = \sum_{i=1}^{n} w_i \cdot x_i. \tag{3.1}$$

At this step the activation function is applied to the weighted sum in order to calculate the output and the weighted sum is compared with a threshold θ to produce an output y that is either 0 or 1, depending on whether or not it exceeds the threshold. Thus,

$$y = \sigma(\varepsilon) = \begin{cases} 1, & \text{if } \varepsilon \geq \theta, \\ 0, & \text{if } \varepsilon < \theta. \end{cases} \tag{3.2}$$

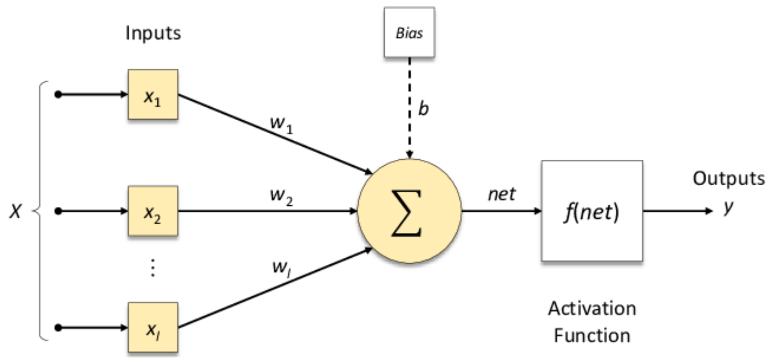


Figure 3.3: The Perceptron [64].

On the other hand, (3.1) and (3.2) can be transformed as follows:

$$\varepsilon = w \cdot x = \sum_{i=0}^{n} w_i \cdot x_i, \tag{3.3}$$

$$y = \sigma(\varepsilon) = \begin{cases} 1, & \text{if } \varepsilon \geq 0, \\ 0, & \text{if } \varepsilon < 0, \end{cases} \tag{3.4}$$

where w and x are *extended weight vector* and *extended input vector* of the perceptron, respectively. Extended weight vector $w = (w_0, w_1, w_2, \ldots, w_n)$ is the vector of weights $w = (w_1, w_2, \ldots, w_n)$ with a bias weight $w_0 = -\theta$. Similarly, the extended input vector $x = (x_0, x_1, x_2, \ldots, x_n)$ is the input vector $x = (x_1, x_2, \ldots, x_n)$ with a bias value $x_0 = 1$.

The structure of a perceptron can be seen in Figure 3.3, where $b = x_0 \times w_0$ is the bias.

3.2.3 Activation Function

The activation function plays very significant role in artificial neural network model architecture. The activation functions do the nonlinear transformation to the input signal in order to make it capable to learn and execute more complicated nonlinear tasks.

On the whole, the nonlinear activation functions are used in neural networks. The most common used types of activation functions are sigmoid, hyperbolic tangent (tanh), and Rectified Linear Unit (ReLU) functions.

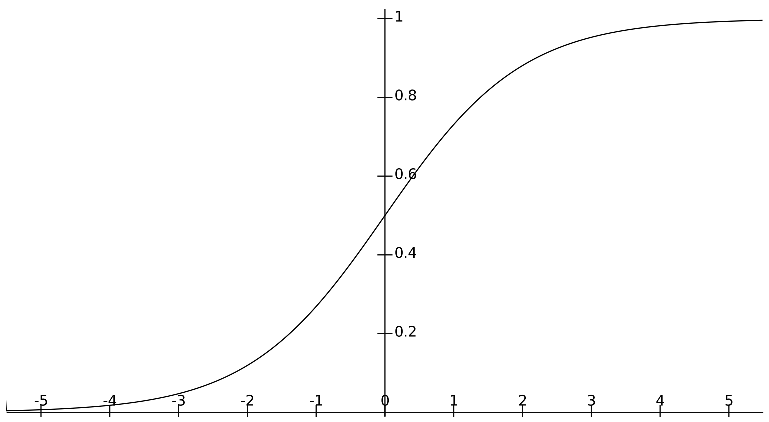


Figure 3.4: Standard Sigmoid function.

The Sigmoid function is one of the most widely used activation functions and sometimes referred to as the logistic function. It is a monotonically increasing function which is defined as:

$$\sigma(x) = \text{sigmoid}(x) = \frac{1}{1+e^{-x}}. \tag{3.5}$$

The sigmoid function takes a real-valued number and transforms it into the range between 0 and 1 (see Figure 3.4). Therefore, it has nice interpretation for output neuron that perform classification task. However, there are some drawbacks of using the sigmoid function due to the vanishing or being small of its gradient values near saturation points, either tail of 0 or 1. The network rejected to learn further or will be remarkably slow.

The hyperbolic tangent (tanh) function is another type of activation functions. It is a nonlinear S – shaped function as the sigmoid function. The main difference between them is that the output range of tanh function is zero-centered, [-1,1] instead of [0,1] (see Figure 3.5). Therefore, the hyperbolic tangent function is more preferred in practice and is given by:

$$\sigma(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}. \tag{3.6}$$

Rectified Linear Unit (ReLU) functions mathematically expressed in (3.7), squashes the net input to a value greater or equal than zero by setting the negative input values to zero (see Figure 3.6). In comparison with sigmoid and tanh function, ReLU computations are cheaper: there is no need for computing the exponential function in activations, and sparsity can be exploited [23]. The advantages of using ReLU in neu-

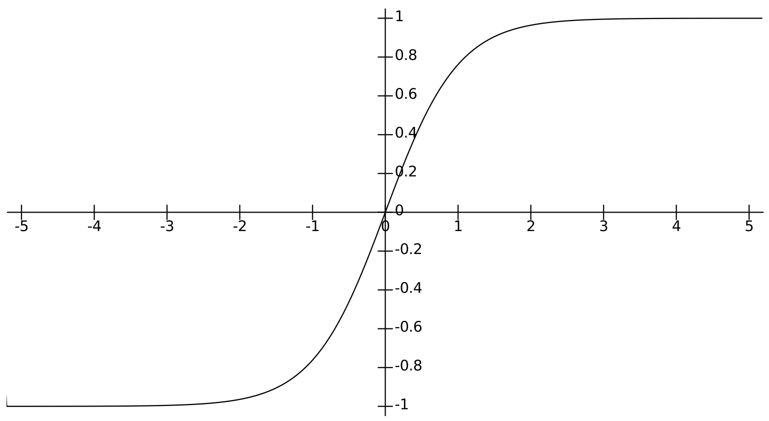


Figure 3.5: Standard Hyperbolic tangent function.

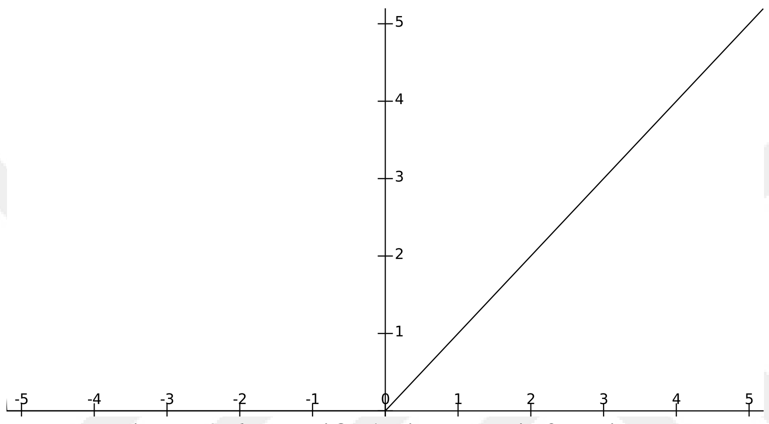


Figure 3.6: Rectified Linear Unit function.

ral networks: it is faster to converge and it doesn't face gradient vanishing problem.

$$\sigma(x) = \text{ReLU}(x) = \max(0, x). \tag{3.7}$$

3.2.4 Loss Function

To determine the capacity of a prediction of a machine learning or statistical model is based on a loss function which is also called as cost or objective function. The main concept of the loss function is to measure the error rate between the predicted and correct target values. Therefore, to receive the best-performed machine learning model, the output of the loss function should be minimized. There are various types of loss functions. Furthermore, these types of loss functions are usually used with specific activation functions in preference.

The Mean-Squared Error (MSE) is the one of the most commonly used loss function that is used to calculate the average squared difference between the predicted and the actual target values:

$$\text{MSE}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2. \tag{3.8}$$

where N is the number of training examples, $\hat{y}_i$ is the model's prediction value and y_i is actual expected output.

The Mean-Squared Error is generally used with the hyperbolic tangent and linear activation functions. In addition, it assumes that the errors are normally distributed.

Another loss function that widely used to measure the error rate in statistical model is Cross Entropy (CE) which is given by the following equation:

$$\text{CE}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i). \tag{3.9}$$

Binary cross entropy is a loss function used on problems involving binary decisions:

$$\text{BCE}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i)(1 - \log(\hat{y}_i))). \tag{3.10}$$

Accordingly, Cross Entropy is used when the activation nodes are representing probabilities and generally it is used in combination with sigmoid activation function in neural networks. In practice, CE mostly leads to faster convergence and better results than mean squared error in terms of classification error rates [24].

3.2.5 Backpropagation

Backpropagation algorithm is a supervised learning algorithm that is used for training neural networks. The goal of the algorithm is to optimize the weights so that the neural network would obtain more valuable predictions which can be reached by minimizing the error rate between the predicted and the correct target values. The described algorithm, which has two steps, forward pass and backward pass, would start after initializing the weights randomly.

The forward pass process begins with calculating the output of the perceptron and continues by finding the loss function. Then, in the backward pass process the obtained loss function would be minimized by using the derivatives and the weights would then be updated. One of the most common used optimization algorithms is the *gradient descent* which uses derivatives to stepwise follow the direction of the negative gradient of the loss function. Since the loss function is a function of the activation function $\hat{y} = \sigma(\varepsilon)$ then the chain rule would be used to calculate the derivative of the loss function. The gradient of the loss function, L, with respect to a certain weight w_i is given as follows:

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial w_i}, \tag{3.11}$$

where ε expressed in (3.3).

The weight then is updated by the following equation:

$$w_i^{'} = w_i - \alpha \frac{\partial L}{\partial w_i}, \tag{3.12}$$

where α is known as the learning rate.

This is the main concept of the algorithm which is applied even to neural networks with more hidden layers and large amount of inputs.

3.2.6 Optimization Algorithm: Adam

As discussed above, the weights are updated by an optimization algorithm. There are several techniques for optimization algorithms such as Adam [42], Adagrad [19], RMSProp, and others. In the subsequent sections of this work the Adam optimizer will be used to train the neural network model due to its fast convergence. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods [42].

Adam (Adaptive Moment Estimation) algorithm is an extension of traditional stochastic gradient descent algorithm. In stochastic gradient descent, there is only one learning rate α that updates all weights and stays constant during the training process. In contrast, Adam optimization algorithm computes the adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients v_t like RMSprop, Adam also keeps an exponentially decaying average of past gradients m_t, similar to momentum:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad \text{and} \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{3.13}$$

where m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively [72].

The Adam update rule for any weight θ is then defined as:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t, \tag{3.14}$$

where $\hat{m}_t$ and $\hat{v}_t$ are given by:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad \text{and} \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}. \tag{3.15}$$

Frequently, the values of β_1, β_2, ϵ are 0.9, 0.999, 10^{-8}, respectively.

3.2.7 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are one of the most successful network architectures in the state-of-the-art artificial neural networks. It is principally utilized in deep learning when handling sequenced data. The main difference from traditional feedforward neural networks, where input features are assumed to be independent to each other, is the significant role of time in recurrent neural networks. Furthermore, RNNs are effective tool to capture information over the dimension of time and store it inside the network so that the texts in the input layer are interpreted as a time sequence of the words. In order to achieve desirable results in various machine learning domains, a few versions of recurrent neural networks have been developed such as Long Short-Term Memory [33] and Gated Recurrent Units [15].

The acceptance of feedback connections which can produce past context information is the vital modernization in RNNs compared to feedforward neural networks. In other words, RNNs have its internal memory which gives privilege over the other neural networks especially when applied to natural language processing tasks where the words can entirely be recognized if the context that they appears in is already learned before.

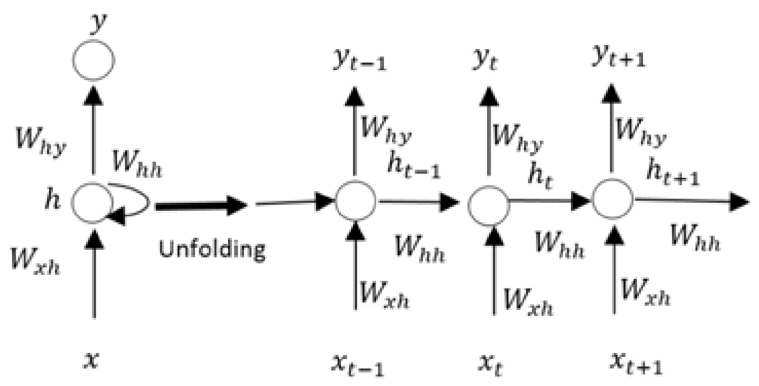


Figure 3.7: General RNN model. Left: folded version of the RNN. Right: unfolded version of the RNN [46].

The basic RNN architecture and its recurrent structure before and after unfolding in time are shown in Figure 3.7. It can be visible that a network with a simple feedback connection after unfolding can be converted into a deep feedforward neural network by adding a new layer in each time step. W_{xh}, W_{hh}, and W_{hy} are the learned weight matrices and x_t, h_t, and y_t are input, hidden state, and output state at time step t respectively.

The forward propagation formulas corresponding to the recurrent neural network models can be described by:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad \text{and} \quad y_t = g(W_{hy}h_t + b_y), \tag{3.16}$$

where b_h and b_y are bias terms; f and g are activation functions where f is generally a nonlinear, differentiable function applied to hidden nodes and g is a function chosen depending on the specific task.

After processing the forward propagation, in order to train RNN models the backward propagation, is needed. Since the RNN model can be represented as a deep feedforward neural network then, the extension of the standard backpropagation algorithm is known as Backpropagation Through Time (BPTT) algorithm [91] is used. The significant distinction is that for every time step the gradients of the weights have to be summed up. Consequently, when the sequences are long the backpropagation process will be very lengthy and cost of computation will be too expensive. In theory, RNNs do not have any limits on using information in arbitrarily long-sequences, but practically standard RNNs can look back only a few steps due to the vanishing or exploding gradient problem [7]. Therefore, there are a number of solutions to this problem such as by truncating the feedback connection after a certain number of time steps, regularization of the RNN's weights [65], training with second order optimization methods [55], and a very careful initialization of RNN's hyperparameters [56]. However, the most effective approach is to use another architecture called Long Short-Term Memory (LSTM) [33].

3.2.8 Long Short-Term Memory

The Long Short-Term Memory (LSTM) unit was initially proposed by Hochreiter and Schmidhuber [33] in 1997. Then, some modifications have been made to the original LSTM architecture by Alex Graves [26]. It is special type of RNNs designed to avoid long-term dependency problem. Unlike the conventional RNNs, the LSTM model capable to remember information for a long period of time [30]. The LSTM networks have been implemented to distinct sequence modelling tasks like machine translation [82], speech recognition [28], with the state-of-the-art performance.

LSTM is made up by a memory cell and its structure is shown in Figure 3.8. The memory block has three main components: input gate, forget gate, and output gate. Essentially, each of the gates has its own responsibility. The input gate decides which information to store at the cell; the forget gate decides which information from the previous hidden state has to be passed to the network; and the output gate controls which information of the new computed hidden state goes to the output vector of the network.

The given formulas below are the mathematical expression of how the network layer in memory cell is updated at each time step t. We denote x_t the vector of the input sequence at time step t and h_t the hidden layer value of the memory cell at time step t. We consider C_t, $\tilde{C}_t$, and C_{t-1} as the current, candidate, and previous cell states, respectively. Firstly, the input and forget vector gates calculated as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad \text{and} \quad f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f). \tag{3.17}$$

Then, the candidate and current cell states are computed:

$$\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad \text{and} \quad C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t. \tag{3.18}$$

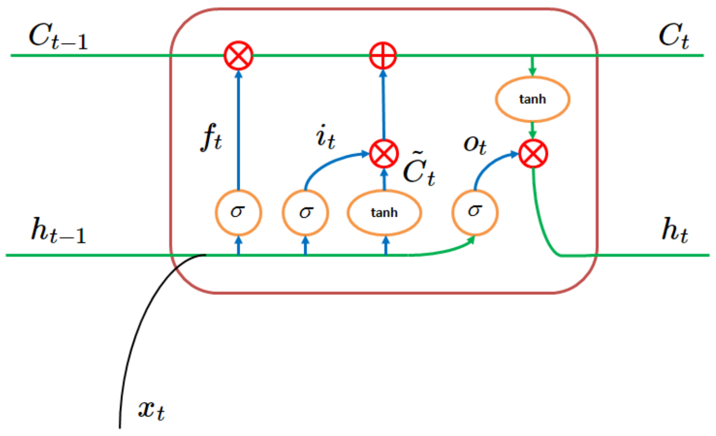


Figure 3.8: LSTM memory block [75].

Finally, the value of output gate and the memory cell output value are calculated:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad \text{and} \quad h_t = o_t \times \tanh(C_t). \tag{3.19}$$

3.2.9 Bidirectional LSTM

The main concept of introducing the bidirectional RNN model is the improvement of standard RNNs performance. While unidirectional RNN use only previous context to predict the next segment for a given sequence, Bidirectional RNN (BRNN) can process both directions with two separate hidden layers where one reads the training sequences forwards, beginning from the start of the sequence, and the other one backwards, beginning from the end of the sequence. Therefore, the BRNN architecture is able to achieve and to use the information from the past and the future states [78].

Bidirectional LSTM (BDLSTM) is received by changing the recurrent neurons of RNNs to the LSTM units and is connected to both hidden layers to the same output. The unfolded structure of this model is presented on Figure 3.9. The output layers of both forward and backward hidden state are computed by using the LSTM updating equations. It has been proved that the bidirectional networks are substantially better than unidirectional ones in many applications, such as phoneme classification [29] and speech recognition [27].

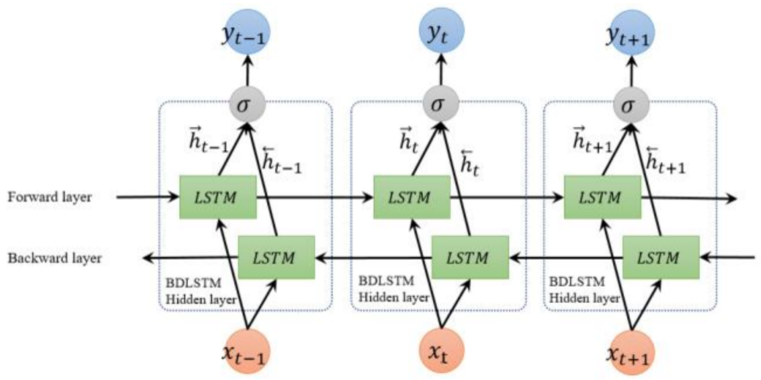


Figure 3.9: Bidirectional LSTM network [16].

3.2.10 Overfitting

Overfitting is a common problem during the learning process of deep learning models. This appears when the model achieves a good fit on the training data; however in new unseen data or validation data, it does not categorize the data correctly. In other words, the model learns and memorizes patterns specific to training data which are irrelevant in validation data. In practice, overfitting is identified by looking at the validation loss or accuracy and by comparing with training loss or accuracy. Generally, the training loss continues decreasing while validation loss in contrast starts increasing after a few number of epochs. To avoid overfitting or to address this issue, there are several remedies such as early stopping, dropout regularization [81], traditional regularizations, reducing the size of the model, getting more data, and batch normalization [35].

During the experiments, the overfitting issue can be seen from the graph that shows the change in accuracy values, calculated on training and validation sets during subsequent iterations of learning process. Moreover, the graph displays from which number of epochs the overfitting has been started. One of the well-known overfitting reduction methods is to interrupt the learning process before the model starts overfitting. This method called *early stopping*. In practice, it allows for a significant improvement in the performance of the model as it demonstrates in this study.

Dropout is a very popular regularization technique mainly applied to reduce overfitting of neural networks. The idea of this method is temporarily dropping neurons during the learning process. The neurons from input and hidden layers are removed randomly from the network and the probability of being kept for each neuron is equal to p. Hyperparameter p is called dropout rate and very often its default value is set to 0.5. Thus, the learning process runs faster since the network becomes smaller.

L2 regularization [60] avoids overfitting by adding the squared magnitude of the weight parameters as a penalty term to the loss function. A term $\lambda \|w\|_2$ is added to the original loss function for each present weight w in the neural network architecture. The regularized version of binary cross entropy from (3.10) is:

$$\text{BCE}_{L2}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i)(1 - \log(\hat{y}_i))) + \frac{\lambda}{2N}\sum_{j=1}^{n_x} w_j^2. \quad (3.20)$$

The choice of λ regularization parameter depends on data and some tuning process is needed. The goal is to catch a proper balance between simplicity and training the data because of the occurring overfitting, if λ is too low, and underfitting, if λ is too high, problems during the learning process of the model. Underfitting appears when a model will not learn enough about the training data to make useful predictions.

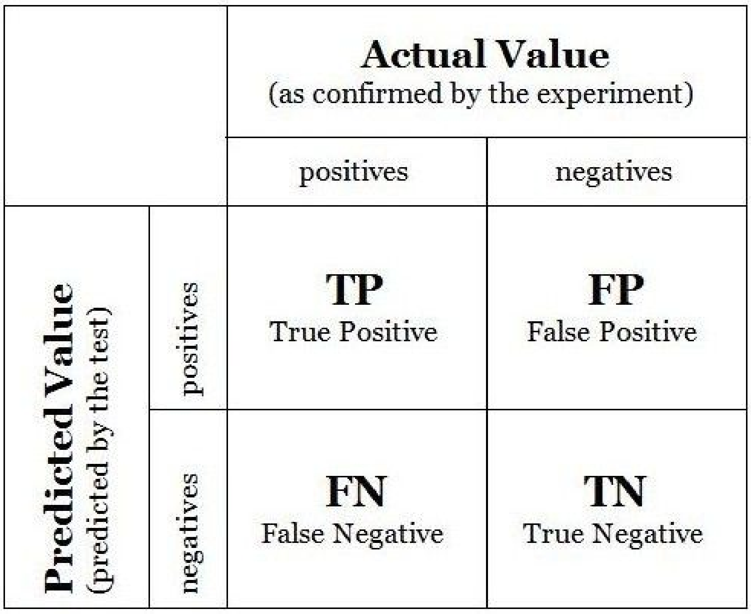
CHAPTER 4

EXPERIMENTAL SETUP

In this section, the datasets used in this work will be described; the setup of the experiments is introduced, and the evaluation metrics to measure the performance of the models are represented.

4.1 Data Description

For the Turkish language, there are not so much available datasets not only for sentiment analysis, but also for other NLP tasks. In 2009, for example, U. Eroğul has created a dataset of Turkish movie reviews for his master thesis manually [20]. In this thesis, we use an available dataset which was created for the paper 'SentiWordNet for New Language: Automatic Translation Approach' [59] and a dataset that is used for U. Eroğul‘s master thesis. A former dataset which has two types of reviews, positive and negative, would be used to demonstrate RNN models for Turkish texts. It has movie reviews which are collected from BeyazPerde [1]. Overall, 53,400 movie reviews by the average length of 33 words are selected. The dataset is well-balanced, that is the number of positive and negative reviews are almost equal. However, for research train and test parts of the dataset are matched as only train in order to split it into new three parts: train, test, and validation set. Moreover, a latter dataset which is created by U. Eroğul would be also used to apply RNN models, distinct methods from earlier implemented machine learning model in U. Eroğul‘s master thesis.



		Actual Value (as confirmed by the experiment)	
		positives	negatives
Predicted Value (predicted by the test)	positives	**TP** True Positive	**FP** False Positive
	negatives	**FN** False Negative	**TN** True Negative

Figure 4.1: Confusion matrix.

4.2 Evaluation Metrics

The choice of an evaluation metric is changeable depending on the task. Generally, various types of evaluation metrics like accuracy, precision, recall, and F1-score are used for sentiment analysis tasks. The mathematical formulas of them are:

$$\text{accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}}, \tag{4.1}$$

$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}}, \tag{4.2}$$

$$\text{recall} = \frac{\text{tp}}{\text{tp} + \text{fn}}, \tag{4.3}$$

$$\text{F1-score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}, \tag{4.4}$$

where true positive (tp) is defined when a positive real value is correctly classified as positive, false positive (fp) is a negative real value which is classified as positive, the true negative (tn) is a negative real value correctly classified as negative, and false negative (fn) is a positive real value that is classified as negative [10]. In Figure 4.1, we depict the confusion matrix.

In this research, the accuracy and F1-score evaluation metrics in (4.1) are used to determine how well they work on sentiment analysis model since they estimate the overall correctness of the system.

4.3 Framework

The programming language Python was used to implement proposed models in this work since it has a large number of scientific libraries for data processing and machine learning algorithms. To perform machine learning or deep learning tasks, there exist several tools or libraries such as TensorFlow [2], Theano [69], Scikit-learn [66], and Keras [13].

Keras is a high level open source neural network API which is written in Python programming language. It is capable of running on top of TensorFlow, Theano or The Microsoft Cognitive Toolkit (CNTK) [79]. It is minimalistic, scalable and support fast experimentation with deep neural networks. Keras is a model-level library, providing high-level building blocks for developing deep learning models. It doesn't handle low-level operations such as tensor manipulation and differentiation. Instead, it relies on a specialized, well-optimized tensor library, serving as the backend engine of Keras [14].

Keras library has been selected for this research since it is so well-suited to the concepts of neural networks and building of simple or complex neural networks takes a few minutes by the help of some powerful models, such as the *Sequential* model and the *Model class used with the functional API*. Therefore, for building neural networks in the research used *Sequential* model of Keras framework whereas for word representation used Python package Gensim [70] in order to handle words to vectors. It is a great package for processing texts and working with word vector models such as *Word2Vec* and *FastText*.

CHAPTER 5

RESULTS AND DISCUSSION

This chapter not only presents the results of experiments for sentiment analysis on Turkish reviews domain but also considers some comparisons with previous work and discussions over the results of experiments for English dataset. The kinds of recurrent neural networks, namely Long Short-Term Memory and bidirectional LSTM, with their hyperparameters are used over two types of datasets to figure out the performance that they could demonstrate. Therefore, the results from the experiments of each model applied with each dataset will be presented separately in following sections. Finally, the last section provides a comparison between described models and the other machine learning technique SVM.

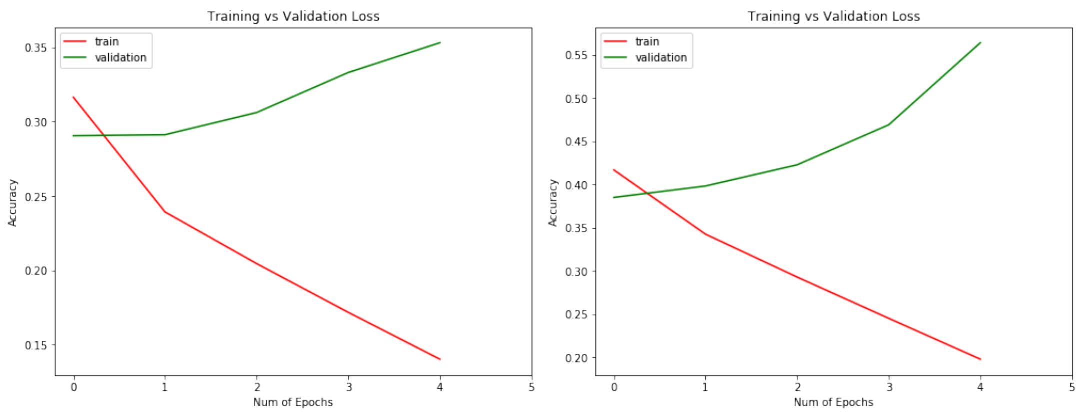
Before start training the model, the first step is to clean data by removing stopwords, punctuations, URLs, and then apply pre-processing techniques such as stemming, lemmatization, tokenization. Zemberek, natural language processing tool for Turkic Languages, is used to handle described processes in this research work. As a consequence, the consecutive step is to map words into vectors that contain numeric values for input to neural network. Word embedding is a type of mapping that allows words with similar meaning to have similar representation. There are several types of word embedding methods including *Word2Vec* and *FastText*, which are the most popular for word representation. In this study, the simple tokenizer under neural networks and most popular word embedding algorithm, *Word2Vec*, is used. Finally, the recurrent neural network architectures are applied to train the model then they are saved and used to predict a new text in order to verify the effectiveness.

In general, the training model could have various numbers of layers but in this research it consists of four layers in total which are Embedding layer, RNN layer, two Dense layers with different input features. For the entire research, we split the main dataset into three sets: training, cross validation, and test set. Since cross validation set is playing main part on getting more truly results of classifiers, the model trained with 20-fold cross validation set. Moreover, Adam optimization algorithm [42] and binary cross entropy loss are used for optimizer and loss function, respectively. For the activation a sigmoid and ReLU functions are used.

To train the model not only the optimization or loss functions but also the batch size and number of epochs are significant, where one epoch is when entire dataset is passed the neural network forward and backward only once and batch size is the total number of training examples presented in a single batch. The reason of dividing one epoch into smaller batches is that it has too large size for calculation at whole. However, the right number of epochs and batch size is unknown and it can vary from domain to domain, from larger dataset to smaller dataset. Generally, the most common used batch size is the 20, 25, and 32. In addition, it is important to use cross validation set as validation data in training model and to shuffle the dataset.

In this research, the fixed number of epochs and batch size are used: 5 and 20, respectively. The reason choosing small number of epochs is to avoid overfitting. More precisely, overfitting is caused even in using just 5 epochs over both of two datasets. In this study, techniques except such as early stopping, dropout regularization, and L2 regularization are used to overcome overfitting. As mentioned earlier, the small number of epochs is used and little size of model is applied. Moreover, in next subsections the results of experiments with and without the regularization techniques are shown and compared.

In this thesis, we use an available dataset which was created for the paper " SentiWordNet for New Language: Automatic Translation Approach " and a dataset that is used for U. Eroğul‘s master thesis. The former dataset is represented as Dataset 1, while the latter is named Dataset 2. A set of positive and negative examples for movie reviews is given in the Appendix A.



(a) LSTM model for Dataset 1 (b) Bidirectional LSTM for Dataset 2

Figure 5.1: An example of overfitting.

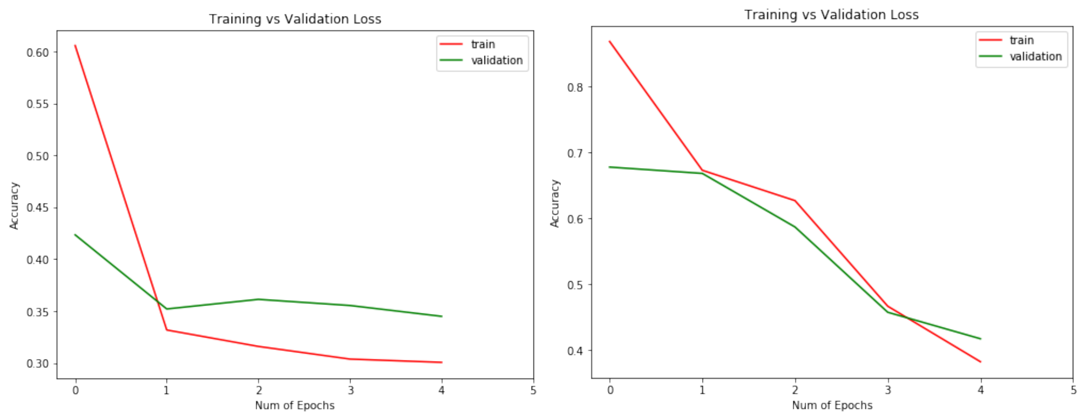
5.1 Results for Experiments with Tokenizer

This specific section only presents the results for experiments that are done by using simple tokenizer under neural networks.

Tokenizer allows to vectorize a text corpus, by turning each text into either a sequence of words ids or into a vector where the coefficient for each token could be binary, based on word count, and based on frequency. Keras provides the Tokenizer class for preparing text documents for deep learning. However, simple Tokenizer has been constructed by ourselves for this research and has been used to tokenize Turkish sentences.

As shown in Figure 5.1, the overfitting problem occurrs on both datasets when applying, both RNN models, LSTM, and bidirectional LSTM. To address the issue dropout regularization and most popular L2 regularization are applied. Despite the fact that the dropout regularization is the powerful tool to address overfitting, in our case it was not the solution to the problem. On the other hand, the L2 regularization technique is the alternative way to solve the problem and to avoid overfitting from the training model. The overfitting problem is addressed using L2 regularization and can be seen in the Figure 5.2.

Table 5.1 presents the results of 20-fold cross-validation of different models on the training set and the results on the test set of a Dataset 1 for movie reviews, while Table 5.2 shows the results for a Dataset 2. The metrics used are the accuracy and



(a) LSTM model with L2 regularization for Dataset 1 (b) Bidirectional LSTM with L2 regularization for Dataset 2

Figure 5.2: An example of avoided overfitting.

F1-score by classes of positive and negative polarities.

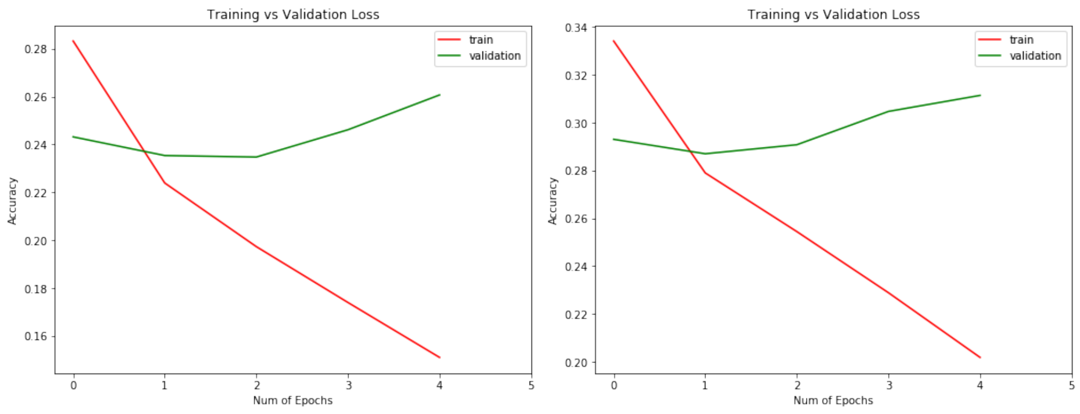
Table 5.1: The results for Dataset 1

DATASET 1	**Validationa data Accuracy**	**Test data F1-score**
LSTM	87.3	87.1
LSTM with Dropout	87.4	87.6
LSTM with L2 regularization	87.8	88.4
Bidirectional LSTM	87	87.3
Bidirectional LSTM with Dropout	87.3	87.3
Bidirectional LSTM with L2 regularization	87.9	88

Table 5.2: The results for Dataset 2

DATASET 2	**Validationa data Accuracy**	**Test data F1-score**
LSTM	81	80.9
LSTM with Dropout	81.4	81.6
LSTM with L2 regularization	81.5	81.7
Bidirectional LSTM	80.5	81.3
Bidirectional LSTM with Dropout	81.2	81.7
Bidirectional LSTM with L2 regularization	81.8	82.7

Since the overfitting is the biggest issue during whole research, the results of the models with L2 regularization are taken as more correctly performance for the datasets. So, LSTM and bidirectional LSTM with L2 regularization show 88.4% and 88% as their F1-score for Dataset 1 while for Dataset 2 these measures are 81.7% and 82.7%,



(a) LSTM model for Dataset 1 (b) Bidirectional LSTM for Dataset 2

Figure 5.3: An example of overfitting.

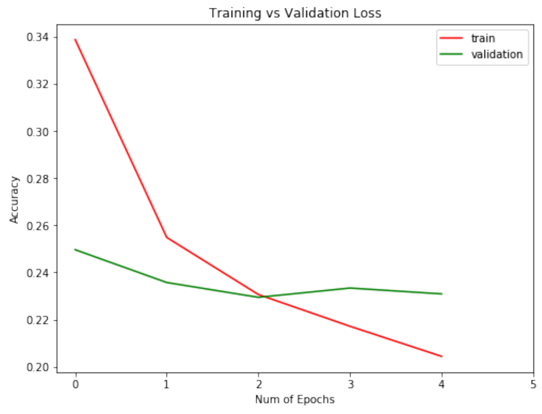
respectively (see the Appendix B). Each of the architectures gives acceptable results for Turkish datasets by using simple tokenizer as a vector representation of words.

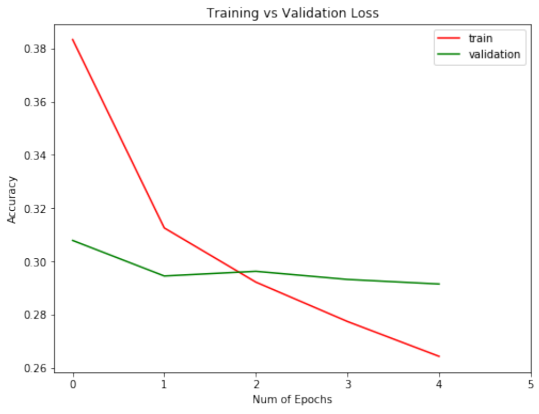
5.2 Results for Experiments with Word2Vec

This section only presents the results for experiments that are done by using state-of-the-art word embedding algorithm, so called *Word2Vec*. For this research, a pre-trained Word2Vec model for Turkish language is utilized.

It can be seen from Figure 5.3 that overfitting starts at some point instead of starting at the beginning point as in the previous section. In such situation to avoid the overfitting, the early stopping method can be used with the epoch where the validation loss starts increasing. However, in this research the regularization methods are also applied to avoid overfitting. The difference from the previous section when used simple tokenizer under neural networks is validation loss, fluctuated when dropout regularization is applied to overcome the overfitting (see Figure 5.4). Finally, the L2 regularization method avoids overfitting from the training model shown in Figure 5.5.

The results for experiments using Word2Vec embedding for Dataset 1 and Dataset 2 can be seen in Table 5.3 and Table 5.4, respectively. The accuracy and F1-score metrics are used for binary classification.

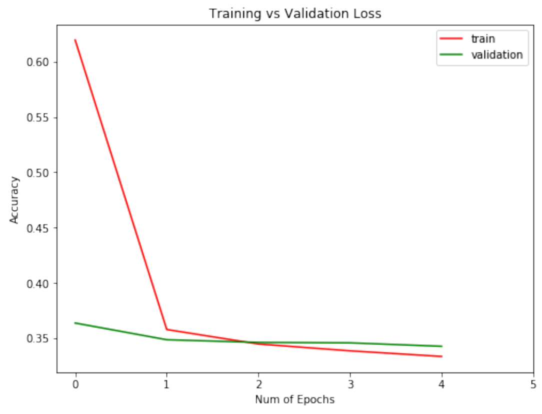


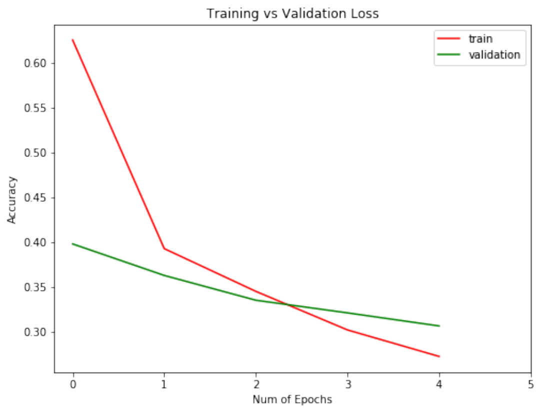


(a) LSTM model with dropout regularization for Dataset 1

(b) Bidirectional LSTM with dropout regularization for Dataset 2

Figure 5.4: An example of fluctuated overfitting.





(a) LSTM model with L2 regularization for Dataset 1

(b) Bidirectional LSTM with L2 regularization for Dataset 2

Figure 5.5: An example of avoided overfitting.

Table 5.3: The results for Dataset 1

DATASET 1	**Validationa data Accuracy**	**Test data F1-score**
LSTM	89.5	84.3
LSTM with Dropout	90.2	85.1
LSTM with L2 regularization	87.5	81.2
Bidirectional LSTM	89.2	83.8
Bidirectional LSTM with Dropout	90	85.1
Bidirectional LSTM with L2 regularization	89.2	83.8

Table 5.4: The results for Dataset 2

DATASET 2	**Validationa data Accuracy**	**Test data F1-score**
LSTM	86.4	78.6
LSTM with Dropout	86.8	79.5
LSTM with L2 regularization	84.9	76.6
Bidirectional LSTM	86.3	78.3
Bidirectional LSTM with Dropout	86.6	79.6
Bidirectional LSTM with L2 regularization	86	78.4

From the tables, it can be seen that depending on the dataset and the model the predicted test can be far worth than the validation or training data. Since overfitting problem cannot be avoided by using dropout regularization, the outcomes made by using L2 regularization would be taken as a result. LSTM architecture gives 81.2% and 76.6% F1-score for Dataset 1 and Dataset 2, respectively, while for the same datasets bidirectional LSTM provides performances of 83.8% and 78.4% for F1-score. To conclude, the bidirectional LSTM overcomes the simple LSTM model in most cases.

5.3 Comparison of the Models

This section provides the comparison between our study and Eroğul‘s master thesis [20] since both studies used the same dataset.

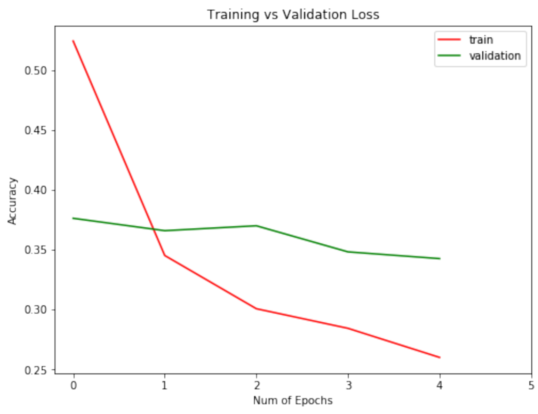
U. Eroğul in his thesis used traditional machine learning algorithm SVM with bag-of-words method and acquired a F1-score of 85% on binary sentiment classification at the document level. In order to compare the results of our work and Eroğul‘s study, only the performances for Dataset 2 is taken from the previous sections. The results are presented in Table 5.5.

Table 5.5: The comparison of the results for Dataset 2

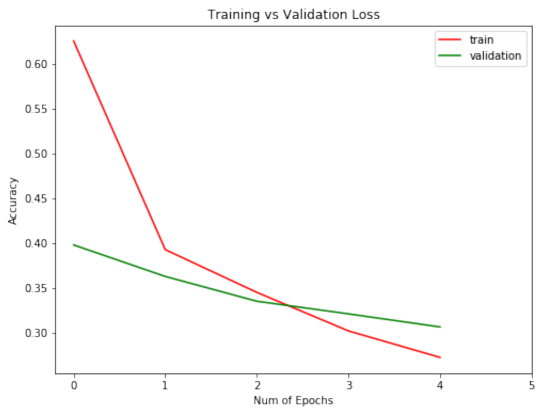
DATASET 2	**Validationa data Accuracy**	**Test data F1-score**
LSTM with L2 regularization using simple tokenizer	81.5	81.7
Bidirectional LSTM with L2 regularization using simple tokenizer	81.8	82.7
LSTM with L2 regularization using Word2Vec	84.9	76.6
Bidirectional LSTM with L2 regularization using Word2Vec	86	78.4
SVM with bag-of-words	85.1	84.9

Looking at the results in Table 5.5, the best result was obtained using SVM algorithm with bag-of-words method. Initially, U. Eroğul acquired a F1-score of 84% by using only the roots of words as features for machine learning and with adding standard spellchecking methods the performance F1-score increased up to 85%. The better result from our study for Dataset 2 is acquired applying bidirectional LSTM with L2 regularization using simple tokenizer and shows good measure of 82.7% as its F1-score. However, models with Word2Vec achieve lower results as compared to models with simple tokenizer or SVM model with bag-of-words due to lower capacity of pre-trained Turkish Word2Vec model.

Finally, we use LSTM and bidirectional LSTM models to perform sentiment analysis on English movie reviews from the Internet Movie Database (IMDB). Keras library is used to build RNN models and its built-in IMDB movie reviews dataset. It means that we do not need any tokenizers for tokenization process since Keras provides tokenized version of IMDB movie reviews. The IMDB dataset was first proposed in [54], as a benchmark for sentiment analysis. As for Turkish dataset, we use the same activation, loss functions and Adam optimizer during the learning process of the model. However, the batch size and number of epochs are changed to 64 and 4, respectively. The reason of choosing such small number of epochs is again due to overfitting problem. Early stopping, dropout, and L2 regularization techniques are applied to address overfitting issue. L2 regularization eventually avoids overfitting



(a) LSTM model with L2 regularization for IMDB dataset



(b) Bidirectional LSTM with L2 regularization for IMDB dataset

Figure 5.6: An example of avoided overfitting.

problem for two models, LSTM and bidirectional LSTM, and can be seen in Figure 5.6.

Table 5.6 presents the results of the models, LSTM and bidirectional LSTM, and also the results from the research [31]. In that study, author achieved excellent results by applying LSTM approach with the word embedding model Word2Vec and showed 95.1% as its F1-score (see the last row in Table 5.6). Since L2 regularization avoids the overfitting problem, then the results with it are taken from our study as more correctly performance for both approaches. So, LSTM and bidirectional LSTM presents performances of 87.7% and 88.2% for F1-score. Therefore, we can see that applying Word2Vec as word representation with the model LSTM increases F1-score by almost 7.5% for English texts.

Table 5.6: The results for IMDB dataset

IMDB dataset	**Validationa data Accuracy**	**Test data F1-score**
LSTM	86.1	86.4
LSTM with Dropout	87.8	87.8
LSTM with L2 regularization	86.9	87.7
Bidirectional LSTM	87	86.8
Bidirectional LSTM with Dropout	88.5	88.9
Bidirectional LSTM with L2 regularization	88.1	88.2
LSTM using Word2Vec		95.1

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, the state-of-the-art recurrent neural networks have been classified for sentiment analysis task on Turkish movie reviews domain. The main concept of using RNN models for this sentiment classification problem is the lack of researches using them for Turkish language. Generally, their architectures provide excellent results for English language and this research encourages this fact since it achieves reasonably good results for Turkish language, too. Furthermore, word representations play a key role in classifying sentiment analysis tasks and choosing the right word representations are also important to achieve acceptable results.

In the first part of research, the simple tokenizer under neural network has been used as a word representation and presents more superior results than the word embedding model Word2Vec. In general, Word2Vec model performs outstanding results for English texts but in this research it achieves reduced results as compared to the simple tokenizer for the same datasets. It can be explained by the lower capacity of pre-trained Turkish Word2Vec model. In spite of this, both word representations with RNN models present results between 81% and 88% for Dataset 1 and the results between 77.6% and 82.7% for Dataset 2, which are preferably results for Turkish domains.

Despite the fact that the models perform well, there can be done more research on changing and reducing the size of the model and taking another batch size to train the model should be carried out for further studies. Moreover, it is known that there exist several types of loss functions, optimizations algorithms which can be adapted and the vocabulary size which can be shaped to a smaller dimension. Therefore, all these changes tend to do more research and achieve valuable results for sentiment analysis tasks on Turkish review domains.

As future work, the new, well-balanced dataset with larger size must be created in order to train deep learning models more precisely for Turkish reviews domain due to the fact that for deep learning models the extension of larger dataset is a significant one. Another approaches different from recurrent neural networks can be implemented to solve sentiment analysis on Turkish texts. More precisely, architectures such as Convolutional Neural Networks (CNNs), FastText for text classification [39], Attention Mechanism [5, 53] and Recursive Neural Networks can be applied since there no works related to these approaches in Turkish.

Different from the approaches mentioned earlier, the word representation is also another important part for improving the performance of models. So, one of the most popular text representation methods is Universal Sentence Encoder[12] representations and its various types. In recent years, these methods applied with deep learning models have become more active in the area of natural language processing. Moreover, another approach is the so called *transfer learning* method [61] which is used earlier for Turkish domains and highly improved the performance of baseline models.

On the whole, since the document level is not just one type of sentiment analysis then the other levels, sentence and aspect, can be a new research work for Recurrent Neural Network architectures.

REFERENCES

[1] Beyazperde, Online Available at: http://www.beyazperde.com.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, Software available from tensorflow.org, 2015.

[3] A. Abbasi, H. Chen, and A. Salem, Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums, ACM Transactions on Information Systems, 26(12), 2008.

[4] A. A. Akın and M. D. Akın, Zemberek, an open source NLP framework for Turkic languages, Structure, 10, pp. 1–5, 2007.

[5] D. Bahdanau, K. Cho, , and Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473, 2014.

[6] F. Benamara, C. Cesarano, A. Picariello, D. Reforgiato, and V. S. Subrahmanian, Sentiment analysis: Adjectives and adverbs are better than adjectives alone, in *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2007.

[7] Y. Bengio, P. Simard, and P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Transactions on Neural Networks 5, pp. 157–166, 1994.

[8] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics 5, pp. 135–146, 2017.

[9] Z. Boynukalın, *Emotion analysis of Turkish texts by using machine learning methods*, Master's thesis, Middle East Technical University, 2012.

[10] A. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognition, 30(7), pp. 1145–1159, 1997.

[11] F. Bre, J. Gimenez, and V. D. Fachinotti, Prediction of wind pressure coefficients on building surfaces using artificial neural networks, Energy and Buildings, 158, 2017.

[12] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. St.John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, and R. Kurzweil, Universal sentence encoder, arXiv preprint arXiv:1803.11175, 2018.

[13] F. Chollet, Keras, https://github.com/fchollet/keras, 2015.

[14] F. Chollet, Deep learning with python, Manning Publications, 2017.

[15] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, Gated feedback recurrent neural networks, in 37, editor, *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pp. 2067–2075, 2015.

[16] Z. Cui, R. Ke, and Y. Wang, Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction, arXiv preprint arXiv:1801.02143, 2018.

[17] K. Dave, S. Lawrence, and D. M. Pennock, Mining the peanut gallery: Opinion extraction and semantic classification of product reviews, in *Proceedings of the 12th International Conference on World Wide Web*, pp. 519–528, 2003.

[18] C. N. Dos Santos and M. Gatti, Deep convolutional neural networks for sentiment analysis of short texts, in *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pp. 69–78, 2014.

[19] J. Duchi, E. Hazan, and Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, The Journal of Machine Learning Research, 12, pp. 2121–2159, 2011.

[20] U. Eroğul, *Sentiment analysis in Turkish*, Master's thesis, Middle East Technical University, 2009.

[21] B. Erşahin, O. Aktaş, D. Kılınç, and M. Erşahin, A hybrid sentiment analysis method for Turkish, Turkish Journal of Electrical Engineering and Computer Sciences, 2019.

[22] G. Gezici and B. Yanikoglu, *Sentiment Analysis in Turkish*, chapter 12, pp. 255–271, Oflazer, K. and Saraçlar, M. (editor). Turkish Natural Language Processing, 2018.

[23] X. Glorot, A. Bordes, and Y. Bengio, Deep sparse rectifier neural networks, in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[24] P. Golik, P. Doetsch, and N. Hermann, Cross-entropy vs. squared error training: a theoretical and experimental comparison, in *14th Annual Conference of the International Speech Communication Association (Interspeech)*, pp. 1756–1760, 2013.

[25] D. Golub and X. He, Character-level question answering with attention, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[26] A. Graves, Generating sequences with recurrent neural networks, arXiv preprint arXiv:1308.0850, 2013.

[27] A. Graves, N. Jaitly, and A. Mohamed, Hybrid speech recognition with deep bidirectional lstm, in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on. IEEE*, pp. 273– 278, 2013.

[28] A. Graves, A. Mohamed, and G. Hinton, Speech recognition with deep recurrent neural networks, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, 2013.

[29] A. Graves and J. Schmidhuber, Framewise phoneme classification with bidirectional lstm and other neural network architectures, Neural Networks, 18(5), pp. 602–610, 2005.

[30] S. Hasim, S. Andrew, and B. Francoise, Long short-term memory recurrent neural network architectures for large scale acoustic modeling, in *15th Annual Conference of the International Speech Communication Association*, 2014.

[31] A. Hassan and A. Mahmood, Sentiment analysis with recurrent neural network and unsupervised neural language model, in *42nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017.

[32] G. Hinton, Learning distributed representations of concepts, in *Proceeding of the 8th Annual Conference of the Cognitive Science Society*, pp. 1–12, 1986.

[33] S. Hochreiter and J. Schmidhuber, Long short- term memory, Neural Computation, pp. 1735–1780, 1997.

[34] W. Hutchins, The Georgetown-IBM experiment demonstrated in January 1954, Conference of the Association for Machine Translation in the Americas, 2004.

[35] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, International Conference on Machine Learning, pp. 448–456, 2015.

[36] V. S. Jagtap and P. Karishma, Analysis of different approaches to sentence-level sentiment classification, International Journal of Scientific Engineering and Technology, 2, pp. 164–170, 2013.

[37] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, in *Proceedings of the 10th European Conference on Machine Learning*, pp. 137–142, 1998.

[38] S. Jose, A three-stage methodological process of data text mining: A ugc business intelligence analysis, Symmetry, 2019.

[39] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, Bag of tricks for efficient text classification, arXiv preprint arXiv:1607.01759, 2016.

[40] M. Kaya, G. Fidan, and I. H. Toroslu, Sentiment analysis of Turkish political news, in *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology*, pp. 174–180, 2012.

[41] Y. Kim, Convolutional neural networks for sentence classification, in *Proceeding of the 2014 Conference on Emprical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, 2014.

[42] D. Kingma and J. Ba, Adam: A method for stochastic optimization, in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[43] M. Z. Kurdi, *Natural Language Processing and Computational Linguistics 2: Semantics, Discourse and Applications*, volume 2, John Wiley and Sons, 2017.

[44] F. Kurt, *Investigating the Effect of Segmentation Methods on Neural Model based Sentiment Analysis on Informal Short Texts in Turkish*, Master's thesis, Middle East Technical University, 2018.

[45] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, Neural architectures for named entity recognition, arXiv preprint arXiv:1603.01360, 2016.

[46] T.-T.-H. Le, J. Kim, and H. Kim, Analyzing effective of activation functions on recurrent network for intrusion detection, Journal of Multimedia Information System, 3, 2016.

[47] Z. Lei and B. Liu, Identifying noun product features that imply opinions, in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT)*, pp. 575–580, 2011.

[48] O. Levy and Y. Goldberg, Dependency-based word embeddings, in 2, editor, *Proceeding of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 302–308, 2014.

[49] C. Lin and Y. He, Joint sentiment/topic model for sentiment analysis, in *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM)*, pp. 375–384, 2009.

[50] B. Liu, *Sentiment Analysis and Subjectivity*, Handbook of Natural Language Processing, Second Edition, 2010.

[51] B. Liu, *Sentiment analysis and opinion mining. Synthesis Lectures on Human Language Technologies*, volume 5, Morgan & Claypool Publishers, 2012.

[52] B. Liu, *Sentiment Analysis Mining Opinions, Sentiments, and Emotions*, Cambridge University Press, 2015.

[53] M.-T. Luong, H. Pham, and C. D. Manning, Effective approaches to attention-based neural machine translation, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1412–1421, 2015.

[54] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Ng, and C. Potts, Learning word vectors for sentiment analysis, in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Association for Computational Linguistics, 2011.

[55] J. Martens and I. Sutskever, Learning recurrent neural networks with hessian-free optimization, in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 1033–1040, 2011.

[56] J. Martens, I. Sutskever, G. Dahl, and G. Hinton, On the importance of initialization and momentum in deep learning, in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 1139–1147, 2013.

[57] A. McCallum and K. Nigam, A comparison of event models for Naive Bayes text classification, in *Proceedings of AAAI-98, Workshop on Learning for Text Categorization*, pp. 41–48, American Association for Artificial Intelligence, 1998.

[58] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, Distributed representations of words and phrases and their compositionality, in *Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013.

[59] B. Naderalvojoud, E. A. Sezer, H. Sever, and A. Ucan, Sentiwordnet for new language: Automatic translation approach, in *12th International Conference on Signal-Image Technology and Internet-Based Systems*, 2016.

[60] A. Ng, Feature selection, l1 vs. l2 regularization, and rotational invariance, in *Proceedings of the International Conference on Machine Learning*, 2004.

[61] S. Pan and Q. Yang, A survey on transfer learning, Knowledge and Data Engineering, IEEE Transactions, 22(10), pp. 1345–1359, 2010.

[62] B. Pang and L. Lee, *Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval*, Now Publishers Inc, 2008.

[63] B. Pang, L. Lee, and S. Vaithyanathan, Thumbs up? : sentiment classification using machine learning techniques, in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 79–86, 2002.

[64] A. Parmezan, V. Alves de Souza, and G. Batista, Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model, Information Sciences, 2019.

[65] R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, arXiv preprint arXiv:1211.5063, 2012.

[66] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in python, The Journal of Machine Learning Research, 12, pp. 2825–2830, 2011.

[67] J. Pennington, R. Socher, and C. D. Manning, Glove: Global vectors for word representation, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Association for Computational Linguistics, Doha, Qatar, 2014.

[68] W. Pitts and W. S. McCulloch, A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, pp. 115–133, 1943.

[69] A. Rami, A. Guillaume, A. Amjad, A. Christof, B. Dzmitry, and B. Nicolas, Theano: A python framework for fast computation of mathematical expressions, arXiv preprint arXiv:1605.02688, 2016.

[70] R. Rehurek and P. Sojka, Software framework for topic modelling with large corpora, in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, 2010.

[71] F. Ronen, Techniques and applications for sentiment analysis, Communications of the ACM, 56, pp. 82–89, 2013.

[72] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747, 2016.

[73] P. Rudy and T. Mike, Sentiment analysis: A combined approach, International Journal of Informatics, 3, pp. 143–157, 2009.

[74] A. M. Rush, S. Chopra, and J. Weston, A neural attention model for abstractive sentence summarization, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[75] J. Said, S. Jadid Abdulkadir, H. Alhussian, M. Nazmi, and A. A Elsheikh, Long short term memory recurrent network for standard and poor's 500 index modelling, International Journal of Engineering and Technology, 7, pp. 25–29, 2018.

[76] G. Salton, A. Wong, and C. S. Yang, A vector space model for automatic indexing, Communications of the ACM, 18, pp. 613–620, 1975.

[77] J. Schultz, How much data is created on the internet each day?, Online Available at: https://www.gwava.com/blog/internet-data-created-daily/, 2012.

[78] M. Schuster and K. Paliwal, Bidirectional recurrent neural networks, IEEE Transactions on Signal Processing, 45(11), pp. 2673–2681, 1997.

[79] F. Seide and A. Agarwal, CNTK: Microsoft's open-source deep-learning toolkit, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2134–2135, 2016.

[80] H. Shirani-Mehr, Applications of deep learning to sentiment analysis of movie reviews, Technical report, Stanford University, 2014.

[81] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research, 15(1), pp. 1929–1958, 2014.

[82] I. Sutskever, O. Vinyals, and Q. Le, Sequence to sequence learning with neural networks, in *Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

[83] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, Lexicon-based methods for sentiment analysis, Computational Linguistics, 37, pp. 267–307, 2011.

[84] M. Thelwall, K. Buckley, and G. Paltoglou, Sentiment strength detection for the social web, Journal of the American Society for Information Science and Technology, 63(1), pp. 163–173, 2012.

[85] P. Turney, Thumbs up or thumbs down? : semantic orientation applied to unsupervised classification of reviews, in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 417–424, 2002.

[86] E. Tutubalina and S. Nikolenko, Demographic prediction based on user reviews about medications, Computacion y Sistemas, 21, pp. 227–241, 2017.

[87] A. Vural, B. Cambazoglu, P. Senkul, and Z. Tokgoz, A framework for sentiment analysis in Turkish: Application to polarity detection of movie reviews in Turkish, Computer and Information Sciences III, pp. 437–445, 2013.

[88] T. Wilson, J. Wiebe, and P. Hoffmann, Recognizing contextual polarity in phrase-level sentiment analysis, in *Proceeding of Conference Human Language and Technology and Empirical Methods in Natural Language Processing (EMNLP)*, pp. 347–354, 2005.

[89] T. Wilson, J. Wiebe, and R. Hwa, Just how mad are you? finding strong and weak opinion clauses, in 4, editor, *Proceeding 19th National Conference of Artificial Intelligence*, pp. 761–769, 2004.

[90] Z. Zhang, D. Miao, and B. Yuan, Context-dependent sentiment classification using antonym pairs and double expansion, in *Web-Age Informative Manage*, pp. 711–722, 2014.

[91] D. Zipser and R. J. Williams, Gradient-based learning algorithms for recurrent networks and their computational complexity, in *Back-propagation: Theory, Architectures and Applications*, pp. 433–486, 1995.

APPENDIX A

EXAMPLE TURKISH DATA

A set of positive and negative movie review examples from each dataset are given in the following sections.

A.1 Positive Movie Reviews

> Aklına sağlık bu filmin yönetmeninin ve yazarının. Bu film hakkında söyleyebileceğim tek şey, hani bi filmle hayatınız bambaşka olur yaa... İzlemeden Ölmeyin!!
>
> Bu filme puan verenleri cok merak ediyorum aslında.her şeyi ile mükemmel bir film 1010 luk bir film ben herşeyi ile 10 verdim.Puan kıranlar acaba neresinden kırıyorlar anlamış değilim.
>
> Edward Norton yeni nesil aktörler içerisinde Johnny deppten sonra belkide en yeteneklisi ve oynadığı tüm filmlerde olduğu gibi sihirbazda da kalitesini konuşturmuş. Ayrıca film yorumlarda sıkca Prestijle kıyaslanmış. Prestij sinema tarihinin en iyi filmlerinden birisi bu filmle kıyaslanılması bence yanlış. Genel olarak bakıldığında film zevkle izlenebilecek seyirciyi sıkmayan kaliteli bir yapım. İzlemenizi tavsiye ederim 10/8

A.2 Negative Movie Reviews

> Tam bir piyasa korku filmiydi ya.. Rezalet yani begenenler bu filmden korkmayi nasil basarmis gerçekten anlamadim.. Vakit kaybetmek istiyorum derseniz yada hangi korku filminde katila katila gülerim diye

sorarsaniz kesinlikle tavsiye ederim.. Ama malesef bundan da kötülerini izledim zamaninda..

Başından sonunun ne olacağını tahmin edebileceğiniz, aşırı derecede durağan, çok fazla belden aşağı muhabbet içeren(çocukları da katan), her üç kelimesinden biri küfür olan bir film olmuş. Gerçekten hayal kırıklığına uğradım. Üstüne üstlük ne Nicholas Cage ne de Michael Caine beklediğim gibiydi ikisi de normalde gösterdikleri performansların o kadar altındaydı ki!!!Boşyere gidip de iki saat sıkıntıdan bayılmayın derim ben....

Fatih akın türkiyenin geleceği parlak isimlerindense türkiye bitmiş desenize.adamın her filminde erotizm her filminde sıkıcı bir psikoloji var. bu filmde öyle boş bir film. farklı duygular katmasını bekliyorsunuz ama hiçbirşey katmıyor haydi kekillinin sahnesi gelsede izlesek diye düşünmekten başka birşey değil bu film.

APPENDIX B

SOURCE CODE

This appendix reports sample codes of the experiment. More precisely, text pre-processing, simple tokenizer, and model construction source codes of the LSTM algorithm with L2 regularization which shows good performance of 88.4% as their F1-score for Dataset 1.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.preprocessing import sequence
from keras.preprocessing.sequence import pad_sequences
np.random.seed(7)

data = pd.read_excel("movie_reviews.xlsx")

from zemberek_parser_master.zemberek_python
import main_libs_my_version as ml
zemberek_api = ml.zemberek_api(
    libjvmpath="C:\\Program Files\\Java\\jre1.8.0_144\\
                                bin\\server\\jvm.dll",
    zemberekJarpath= "C:\\Users\\admin\\Documents\\
                                zemberek_parser_master\\
    zemberek_python\\zemberek-tum-2.0.jar").zemberek()

X_data = pad_sequences(reviews, maxlen = 50)
y_data = np.asarray(labels)

X_train, X_test, y_train, y_test =
        train_test_split(np.asarray(X_data), np.asarray(y_data),
                                                 test_size = 0.2)
X_train, X_val, y_train, y_val =
        train_test_split(X_train, y_train, test_size=0.2,
                                                 random_state=42)
```

Listing B.1: Text preprocessing source code in Python.

```python
reviews  = []
labels = []
all_tokens  = []
unique_tokens = dict()
for i in range(len(data)):
    try:
        tokens = ml.ZemberekTool(zemberek_api).
                    metinde_gecen_kokleri_bul(data["Text"][i])
        reviews.append(tokens)
        labels.append(data["Sentiment"][i])
        all_tokens += tokens
        for t in tokens:
            if t in unique_tokens.keys():
                unique_tokens[t] += 1
            else:
                unique_tokens[t] = 1
    except:
        pass
def create_dictionary(unique_tokens, threshold):
    token_to_idx = dict()
    idx_to_token = dict()
    unique_token_keys = list(unique_tokens.keys())
    j = 0
    for i in range(len(unique_token_keys)):
        if unique_tokens[unique_token_keys[i]] > threshold:
            token_to_idx[unique_token_keys[i]] = j
            idx_to_token[j] = unique_token_keys[i]
            j += 1
    return token_to_idx, idx_to_token
token_to_idx, idx_to_token = create_dictionary(unique_tokens, 30)
for i in range(len(reviews)):
    for j in range(len(reviews[i])):
        if reviews[i][j] in token_to_idx.keys():
            reviews[i][j] = token_to_idx[reviews[i][j]]
        else:
            reviews[i][j] = None
    reviews[i] = [x for x in reviews[i] if x != None]
i = 0
for r in reviews:
    if len(r)!=0:
        i += 1
for i in range(len(reviews)):
    if len(reviews[i]) == 0:
        labels[i] = None
reviews = [x for x in reviews if len(x) != 0]
labels = [x for x in labels if x != None]
```

Listing B.2: Simple tokenizer source code in Python.

```
import numpy as np
import pandas as pd
from keras.models import Sequential, load_model
from keras.layers
import Dense, Dropout, Activation, LSTM, Embedding, Bidirectional
from keras.layers.embeddings import Embedding
from keras.regularizers import l2
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report, confusion_matrix

embedding_size=400
model = Sequential()
model.add(Embedding(len(token_to_idx), embedding_size,
                    input_length = X_train.shape[1]))
model.add(LSTM(50,kernel_regularizer=l2(0.01),
               recurrent_regularizer=l2(0.01),
               bias_regularizer=l2(0.01)))
model.add(Dense(50, activation = "relu",kernel_regularizer=l2(0.01),
                bias_regularizer=l2(0.01)))
model.add(Dense(1, activation = "sigmoid"))
print(model.summary())

nb_epoch = 5
batch_size = 20
model.compile(loss = "binary_crossentropy",
              optimizer = "adam", metrics = ["accuracy"])
hist_lstm = model.fit(X_train, y_train,
                      validation_data=(X_val, y_val),shuffle=True,
                      epochs = nb_epoch, batch_size = batch_size)
score = model.evaluate(X_val, y_val)
print("Validation Loss: %.2f%%" % (score[0]*100))
print("Validation Accuracy: %.2f%%" % (score[1]*100))
model.save('lstm_regularizer_movie_reviews.h5')

y_pred = model.predict(X_test)
yy_scores = (y_pred > 0.5)
yy_true = (y_test>0.5)
print("F1 Score: " + str(f1_score(yy_true, yy_scores,
                                  average='weighted')))
print(confusion_matrix(y_test, yy_scores))
print(classification_report(y_test,yy_scores))
```

Listing B.3: A LSTM model source code in Keras.