INVESTIGATING THE PERFORMANCE OF SEGMENTATION METHODS WITH DEEP LEARNING MODELS FOR SENTIMENT ANALYSIS ON TURKISH INFORMAL TEXTS

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

FATİH KURT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE
IN
INFORMATION SYSTEMS

JANUARY 2018

Approval of the thesis:

INVESTIGATING THE PERFORMANCE OF SEGMENTATION METHODS WITH DEEP LEARNING MODELS FOR SENTIMENT ANALYSIS ON TURKISH INFORMAL TEXTS

submitted by **FATİH KURT** in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems Department, Middle East Technical University** by,

ABSTRACT

INVESTIGATING THE PERFORMANCE OF SEGMENTATION METHODS WITH DEEP LEARNING MODELS FOR SENTIMENT ANALYSIS ON TURKISH INFORMAL TEXTS

Kurt, Fatih

M.S., Department of Information Systems

Supervisor : Prof. Dr. Pınar Karagöz

January 2018, 65 pages

This work investigates segmentation approaches for informal short texts in morphologically rich languages in order to effectively classify the sentiment. The two building blocks of the proposed work in this thesis are segmentation and deep neural network model building. Segmentation focuses on preprocessing of text with different methodologies. These methodologies are grouped under four distinct approaches; namely, morphological, sub-word, tokenization, and hybrid approaches. There is mostly multiple numbers of variants for each of these four methods provided in this work. The second stage focuses on effective model building for classifying text. Performances of each method are evaluated by utilizing a model built by a Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) model proposed in the literature for text classification.

Keywords: Sentiment Analysis, Deep Learning, Neural Networks, NLP for User-generated Content, Word Segmentation

ÖZ

SEGMENTASYON YÖNTEMLERİNİN INFORMAL TÜRKÇE METİNLERDE DUYGUSAL ANALİZ İÇİN DERİN ÖĞRENME MODELLERİ İLE KULLANIMINDA PERFORMANS ANALİZİ

Kurt, Fatih
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi : Prof. Dr. Pınar Karagöz

Ocak 2018 , 65 sayfa

Bu çalışma morfolojik olarak zengin dillerde kısa informal metinlerde etkili bir şekilde duygu analizi yapılmasını incelemektedir. Bu tezde önerilen çalışmanın temel yapı taşlarını metin segmentasyonu ve yapay sinir ağı modeli yaratılması oluşturmaktadır. Segmentasyon farklı metodolojilerle metinlerin ön-işlemden geçirilmesini sağlar. Bu çalışmada kullanılan metodolojiler dört farklı ana groupta kümelenmiştir. Bunlar, morfolojik, kelime-altı sözcük, dizgeleme ve hibrit metodlarından oluşmaktadır. Çoğunlukla bu ana grupların altında birden fazla farklı metod kullanılmıştır. İkinci aşama metinlerin sınıflandırılması için etkili model yaratmaya odaklanmadtadır. İlk aşamada oluşturulan herbir segmentasyon yöntemi için Konvolüsyonel Sinir Ağı (KSA ya da CNN) ve Tekrarlayan Sinir Ağı (TSA ya da RNN) modellerinin litaratürde iyi bilinen parametreleri kullanılarak performans testleri yapılmaktadır.

Anahtar Kelimeler: Duygu Analizi, Derin Öğrenme, Sinir Ağları, Kullanıcı içerikleri için Doğal Dil İşleme, Kelime Segmentasyonu

To my family and valuable friends

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Dr. Pınar Karagöz for her constant support, guidance, and friendship. It was a great honour to work with her for the last two years and our cooperation influenced my academical and world-view highly.

A lot of people influenced and supported this work scientifically and their contributions were most valuable for me. Dilek Önal personally pushed for the very idea of the thesis and provided further interesting and valuable ideas on how to do follow-up work regarding the thesis. She also supplied a lot of important material for the real kick off of this work. Her ideas and support made it possible that in a short time I was able to build the frame of this work.

I would also like to thank all people around me that contributed in providing valuable information and taking part in discussions that eventually made this work better.

And there are a lot of people that were with me in these years. They defined me, they made me who I am, they are true owners of this work. It is not possible to write down why each of them is important to me and this work because it will take more space than the work itself.

Very special thanks to a group of people who've taught me real meaning of friendship lately. I would like to extend my deepest gratitude to Arda Taşcı, Deniz Çelik, and Fatih Hafızoğlu for their constant support during this thesis and in my life in general.

Special thanks to all good people around me, for being good friends and showing their support any time in my life. I would like to thank Arda güçlü, Barış Ercan, Burçak Gündoğdu, Oğuz Kerem Şengöz, Onur Yılmaz, and many others that I could not list here.

Many thanks to my company Arçelik A.Ş., my team leader Deniz Kaya, and my other managers in the company not only for making the postgraduate education possible during work but also for actively encouraging it and providing much-needed assistance and flexibility.

And finally, I would like to thank my family for their life-long support, for being with me all the time, and for making me what I am today. There can be no other group of people that contributes to one's life, its meaning and meritorious values more than one's family. There is no adequate word that can describe my love and gratitude for them.

TABLE OF CONTENTS

ABSTRACT vi
ÖZ vii
ACKNOWLEDGMENTS ix
TABLE OF CONTENTS x
LIST OF TABLES xii
LIST OF FIGURES xiv
LIST OF ABBREVIATIONS xvi

CHAPTERS

1 INTRODUCTION **1**
2 RELATED WORK **5**
2.1 Neural networks for Sentiment Analysis 5
2.2 Sentiment Analysis for Turkish 6
3 SENTIMENT ANALYSIS WITH VARIOUS SEGMENTATION METHODS AND DEEP LEARNING MODELS **9**
3.1 Segmentation Methods 9
3.1.1 *Word-based* Analysis 10
3.1.2 *Morphology-based* Analysis 10
3.1.3 *Sub-word* Analysis 12
3.1.4 *Hybrid* Analysis 15
3.2 Deep Neural Networks 16
3.2.1 Convolutional Neural Networks 16
3.2.2 CNN-rand 16
3.2.3 CNN-rand Simplified 19
3.2.4 Long-Short Term Memory (Neural Network) 21
3.2.5 Proposed LSTM Model 22
3.2.6 Hyper-parameter Optimizations 23
3.2.7 Model Training Callbacks 25
3.3 Limitations 26
4 EXPERIMENTS **29**
4.1 Data Sets 29

4.2 Data Pre-processing 30
4.2.1 Review Sentence Break-down 32
4.2.2 Forced Review Break-down 33
4.2.3 Sentiment Loss with Review Break-down and Majority Voting . 34
4.2.4 Segmentation 34
4.2.5 Token Identification 34
4.2.6 Sentence-Level Review Modeling for Train-Test Split 36
4.3 Deep Neural Networks 38
4.4 Baseline Methods 39
4.5 Experiment Results and Discussion 40
4.5.1 CNN-rand Results 41
4.5.2 Simplified CNN-rand Results 41
4.5.3 LSTM Results 43
4.5.4 Results in Comparison to Baseline Scores 43
4.6 Performance Evaluation 46
4.6.1 Distribution of Polarity Predictions and Majority Voting 47
4.6.2 Performance Comparison on Sample Cases 49
4.6.3 Cross-parameter Comparison 51
4.6.4 Computational Performance 52
4.7 Summary 55
5 CONCLUSION AND FUTURE WORK 57

LIST OF TABLES

Table 1.1 Turkish words w/ and w/o negation 2

Table 3.1 Running examples for *Word-based* segmentation methods 10
Table 3.2 Running examples for *Morphology-based* segmentation methods . 11
Table 3.3 Running examples for *Sub-word* segmentation methods 14
Table 3.4 Running examples for *Hybrid* segmentation methods 15

Table 4.1 *Movie Reviews* initial statistics 29
Table 4.2 *Product Reviews* initial statistics 30
Table 4.3 Vocabulary sizes and other basic statistics for *Movie Reviews* and *Product Reviews* datasets. 30
Table 4.4 *Movie Reviews* and *Product Reviews* post-split stats 32
Table 4.5 *Movie Reviews* and *Product Reviews* maximum sentence size after forced split 34
Table 4.6 Compiled Vocabulary sizes and Id counts for each segmentation methods 35
Table 4.7 Number of sentences in each sample after training/validation/test dataset split. 37
Table 4.8 Baseline Scores on the *Movie Reviews* data set 39
Table 4.9 Baseline Scores on the *Product Reviews* data set 40
Table 4.10 CNN-rand Accuracy Results w/o Majority Voting 41
Table 4.11 CNN-rand Accuracy Results w/ Majority Voting 42
Table 4.12 Simplified CNN-rand Accuracy Results w/o Majority Voting . . 42
Table 4.13 Simplified CNN-rand Accuracy Results w/ Majority Voting . . . 43
Table 4.14 LSTM Accuracy Results w/o Majority Voting 44
Table 4.15 LSTM Accuracy Results w/ Majority Voting 44
Table 4.16 Accuracy Scores on the *Movie Reviews* data set 45
Table 4.17 Accuracy Scores on the *Product Reviews* data set 46
Table 4.18 Margin between best baseline scores and our results on *Movie Reviews*, and *Product Reviews* data sets 46

Table 4.19 Sample Predictions for a review compiled from LSTM @ *BPE-5k* and LSTM @ *Lemma+Suffix* 50

Table 4.20 Sample Predictions for a review compiled from CNN @ *BPE-5k* and CNN @ *Lemma+Suffix* 51

Table 4.21 List of parameters collected during experiments with pre-precessing and neural model building. 52

LIST OF FIGURES

Figure 3.1 Figure showing a potential utilization Convolutional Neural Networks for text classification job (Kim, 2014) 17

Figure 3.2 CNN-rand network described in Kim (2014) with layer shapes produced for *Word-Token* segmentation on Movie Reviews dataset . . 17

Figure 3.3 Simplified CNN-rand network proposed by alexander-rakhlin with layer shapes produced for *Word-Token* segmentation on Movie Reviews dataset 20

Figure 3.4 An LSTM cell detailed in Graves et al. (2013) with input, output, and forget gates. 21

Figure 3.5 A simple LSTM network with layer shapes produced for *Word-Token* segmentation on Movie Reviews dataset 22

Figure 3.6 Filter sizes used for Grid Search 24

Figure 3.7 Accuracy graph for Dropout broken down by Filters. Color shows Accuracy. Shape shows details about L2 Regularization. 25

Figure 4.1 Histogram showing the distribution of review length for datasets. 31

Figure 4.2 Histogram showing the distribution of review length for datasets after review-sentence-split. 33

Figure 4.3 Word-token frequency distribution over percentile. Y-axis is limited to 10 in order to show lower frequencies better. 36

Figure 4.4 CNN-rand: Progress of accuracy and loss for validation and training data on Movie dataset w/ *BPE-5k* segmentation. The green vertical line shows the position of best model save, while orange one shows patience count-down for EarlyStopping. 38

Figure 4.5 Predicted polarity distribution of test sample data for an LSTM networks trained with *BPE-5k* segmentation output. Distribution is Sentence-level. 47

Figure 4.6 Predicted polarity distribution of test sample data for an LSTM networks trained with *BPE-5k* segmentation output. Distribution is Review-level 48

Figure 4.7 Predicted polarity distribution of test sample data for a CNN networks trained with *BPE-5k* segmentation output. Distribution is Sentence-level 49

Figure 4.8 Predicted polarity distribution of test sample data for a CNN networks trained with *BPE-5k* segmentation output. Distribution is Review-level 50

Figure 4.9 Scatter plot showing relation between vocabulary size and accuracy among all results acquired with CNN. Review length and Segmentation are added as details. 53

Figure 4.10 Bar chart showing average memory size, average test data prediction duration, and model memory for a CNN network. 55

LIST OF ABBREVIATIONS

BPE	Byte-pair encoding
BPE-1k	Byte-pair encoding with 1k vocabulary
BPE-5k	Byte-pair encoding with 5k vocabulary
BPE-30k	Byte-pair encoding with 30k vocabulary
CNN or ConvNet	Convolutional Neural Network
CNN-rand	A special implementation of CNN with random initial embedding
DCNN	Dynamic Convolutional Neural Network
DNN	Deep Neural Network
GPU	Graphics Processing Unit
IMDB	Internet Movie Database: Online Movie Review Web-site
K-means	A machine learning clustering model
LSTM	Long-Short Term Memory (Neural Network) - A special type of RNN
ML	Machine Learning
MV-RNN	Matrix-Vector Model for RNN
NB	Naive Bayes
NER	Named Entity Recognition
NLP	Natural Language Processing
n-gram	An n-gram is a contiguous sequence of n items from a given sample of text or speech
RNN	Recurrent or Recursive Neural Network
SVM	Support Vector Machine
Word-net	Lexical database for the English language

CHAPTER 1

INTRODUCTION

Sentiment analysis, being a text classification task, has been a popular topic in natural language processing related academic works, which has been studied by a high number of researchers. As also mentioned by Giachanou in Giachanou and Crestani (2016) social media data created by a huge number of individual users is very dynamic and varies to a great extent both in terms of size and type. The direct interaction of users with different products, organizations, current events, and people urges involved actors to create means to automate the understanding of user sentiment on a particular topic. Sentiment analysis being more or less about understanding author feeling, it largely focuses on positive or negative sentiment directed at an actor or an event. Earlier techniques on sentiment extraction have focused on the lexicon and rule-based solutions; however, studies recently started to employ Neural Networks. Even though latest studies show promising results, (Yin et al., 2017) they largely focus on English and lack in-depth investigation for other languages.

There are two major challenges with most languages. First is that they mostly lack well-known and useful resources such as Word-net, and there are few types of research on them. The other is that some of the languages are morphologically very rich, which leads to a different set of problems.

Lack of NLP resources such as WordNet, (Miller, 1995) SentiWordNet (Esuli and Sebastiani, 2006; Baccianella et al., 2010), and SenticNet (Cambria et al., 2014) is a problem in case of lexicon or rule-based approaches being used. However, usage of neural network approaches based on embeddings and representation learning can eliminate this effect to a great extent. Resources such as WordNet helps neural networks in terms of providing initial vector representations for each token that will be processed by the network. However, neural networks are capable of building their embedding layer with the gradually built-up vector representations during the training period. Starting neural network with random embeddings instead of with preset vector representations will only extend the train duration assuming the dataset used is rich enough.

One has to mention that the scheme explained in the previous paragraph about network building its own embedding is proved to be effective in English but not in Turkish, where the morphologically rich structure brings the large vocabulary problem. A large vocabulary will effectively diminish neural networks effectiveness in extracting proper vector representations for words and building the corresponding embedding later. This is partly due to large vocabulary having the majority of tokens with

Table 1.1: Turkish words w/ and w/o negation

Turkish	**Translation**
Seviyorum	I like
Sev**mi**yorum	I don't like
Konuşmaktayız	We are (in the act of) talking
Konuş**ma**yız	We don't talk

very small frequencies within the text, and partly due to different variations of words particular affixes being used in certain contexts.

In addition to the large vocabulary problem emanating from the morphological structure of Turkish, the informal texts also contribute to the problem with the introduction of free usage of language with slang, typographical errors, abbreviations, and the local usage of language in different areas. Several prominent approaches that can tackle these problems are presented in this work. Morphological analysis, sub-word segmentation are some of these approaches that can deal with large vocabulary problem and unstructured usage of the language.

In this study, it is aimed to address challenges explained above with plausible approaches.

In morphologically rich languages, token derivation depends heavily on grammar rules and affixes. This has two major consequences that bring additional challenges for sentiment extraction task. Firstly, the language has a very large vocabulary that causes sparsity and a high level of dimensionality when a bag of words representations are used. A large vocabulary is a problem for the neural models as the size of word embedding matrix increases. This causes neural network learn slower and less efficient. Secondly, an affix can determine the entire sentiment of a sentence, thereby increasing the importance of correctly identifying sub-word elements that incorporate such sentiments. Table 1.1 shows good examples of simple suffixes having a major sentimental effect.

As already mentioned before, most languages mostly lack advanced resources such as Word-net and studies that mostly depend on frameworks that are still in the making or already abandoned. One such development is Zemberek, (Ahmet Afsin Akin, 2007) which is originally developed by Ahmet Afsin Akin with further improvements until 2010[1]. The project was discontinued or seen a low level of development traffic for a while; however, it was still being widely used due to lack of resources to utilize otherwise. The project was re-adopted with a new code base[2] starting in late 2013 with further improvements to the date.

In this work, we primarily focus on Turkish. This is because Turkish suffers both from large vocabulary due to its morphology and from lack of resources such as Word-net. Sentiment analysis for Turkish has been addressed using lexicon (Erogul, 2009; Vural

[1] `github.com/ahmetaa/zemberek`

[2] `github.com/ahmetaa/zemberek-nlp`

et al., 2012; Kokciyan et al., 2013; Yildirimm et al., 2015) or rule-based (Boynukalin, 2012; Firat Akba and Sever, 2014; Coban et al., 2015) methods.

To the best of our knowledge, this is the first work that reports an empirical evaluation of a neural network based approach to sentiment analysis for Turkish. Kisa and Karagoz (2015); Demir and Özgür (2014); Kuru et al. (2016) are examples of successful use of word embeddings and neural nets for Named Entity Recognition (NER). Kuru et al. (2016) propose a character-level LSTM for the NER task.

The main contribution of this work is investigating the effect of various segmentation models, including word-based, character-based, as well as morphological analysis based segmentation for sentiment classification. For the sake of completeness to the investigation, several different neural network models are also incorporated into the text classification pipe-lining. Together with the challenges of working with a morphologically rich language, another challenge is the use of social media resources, which enforces no formal grammar or language controls over the accuracy of the input texts. We evaluate each segmentation method with a CNN and an RNN model, which, together are two states of the art neural network models for sentiment analysis. (Yin et al., 2017)

In this work, it is planned to demonstrate literature research, various proposals, and related experiments in order to present valuable results that will contribute in the ways listed below.

- This work will enable to evaluate effects of different segmentation methods on performance in terms of accuracy and computational effectiveness.
- It will also enable to compare different deep neural networks.
- To the best of our knowledge, it will be the first sentiment analysis experiment which applies Neural Network on Turkish data sets.
- Finally, it will also be a novel approach both in terms of comparing different segmentation methods and neural networks models for Turkish.

In this work, we will cover a full investigation into research, planning, and experiments for sentiment analysis in Turkish using Deep Neural Networks. We will present state of the art researches and explain experiments exerted for Sentiment Analysis using different approaches each. We will also discuss different technologies and methodologies used for text segmentation and classification. We will propose a complete methodology to identify performances for different segmentation methods and neural networks in terms of sentiment extraction accuracy and computational effectiveness. We will also present datasets to be used for this work and the source for these datasets. We will also present baseline scores for these datasets by investigating the original works and methodologies exerted in order to acquire these baseline scores.

The organization of this thesis will be as follows.

Previous Work: In this part, we will present state of the art Sentiment analysis researches and experiments both for Turkish and other languages.

Sentiment Analysis with Various Segmentation Methods and Deep Learning Models: In this part, we will present various text segmentation methods and deep learning models, and propose a methodology for Turkish Sentiment Analysis.

Experiments and Results: In this part, we will present our workflow in order to execute processes involved in proposed methodology.

Conclusion and Future Work: Finally, we will evaluate our results and conclude with potential future extensions to this work.

CHAPTER 2

RELATED WORK

2.1 Neural networks for Sentiment Analysis

In recent years, neural network models that can encode the sentiment of a text into a distributed representation have been studied intensively. Besides, studies that rely on pre-trained word embeddings such as the Bag of Semantic Concepts, (Lebret and Collobert, 2014), recursive neural networks (Socher et al., 2012, 2013; Irsoy and Cardie, 2014a,b) and CNNs (Kalchbrenner et al., 2014) have been utilized as neural semantic compositionality models for sentiment. Yin et al. (2017) compare CNN and RNN and their usages in NLP related studies.

In their work, Lebret and Collobert (2014) use bag of words approach by extracting n-gram representations from words. The main motivation for breaking down individual words into smaller sub-components manifested as a way of dealing with large vocabulary size. They also use K-means clustering to further deduct vector size for word representation. By doing so only each cluster will have a vector, which will reduce resource constraints significantly. Throughout work, for n-grams, and K-means $n = \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 2, 3\}$, and $K = 100, 200, 300$ settings are used for IMDB movie dataset, Pang and Lee (2004), and Maas et al. (2011) datasets. On the former dataset, 84.0% accuracy is achieved with 1+2-grams for both 200-means and 300-means. On the latter dataset, 88.55% accuracy is achieved with 1+2+3-grams and for 300-means.

Irsoy and Cardie (2014a) propose using a positional directed acyclic graph with a Recursive Neural Network. (Socher et al., 2011) Since word vectors are built into recursive binary trees, a Recursive Neural Networks are very good at handling them. In this method, a weight for a sentiment is calculated by extracting a weight for each token also using their binary tree and relation to lower nodes with back-propagating their weights.

Another example of using Deep Neural networks for sentiment and opinion extraction is provided by Irsoy and Cardie (2014b). In this work, they propose using an Elman-type Recurrent Neural Network (Elman, 1990) with improved features such as adding neural depth by stacking Elman hidden layers (Hermans and Schrauwen, 2013) and utilizing bi-directionality with Bidirectional RNN architecture proposed by Schuster and Paliwal (1997).

A major problem with word-vectors mentioned by many types of research is the limitation on vector space definitions. (Lebret and Collobert, 2014; Turney and Pantel, 2010) Socher et al. (2012) propose using a Recursive Matrix-Vector Model (MV-RNN) to achieve compositionality. The model enables to learn compositional vector representations for phrases and sentences that constitute various types and lengths. The model achieves this by assigning a vector and a vector matrix to every node in the parse tree. The MV-RNN starts with building multi-word vectors by building multiples of single word vector representations using vectors from constituting words.

Li et al. (2015) compare RNN and recursive neural models on the tasks of sentiment classification of sentences and syntactic phrases, question answering, discourse parsing, and semantic relations. They report that RNN models have an equal or superior performance to recursive models except for the semantic relations between nominals task.

Kalchbrenner et al. propose Dynamic Convolutional Neural Network (DCNN) model (Kalchbrenner et al., 2014) in order to model semantic compositionality of sentences. The DCNN model is compared against recursive neural networks on Stanford Sentiment Tree-bank. (Socher et al., 2013) The DCNN model is shown to outperform the recursive neural network model of Socher et al. (2013) on binary and multi-class sentiment classification. It is important to note that, in their work, DCNN model is not compared with RNN models.

CNN models proposed by Kim (2014) for sentiment analysis on sentences outperform DCNN on 2/3 datasets. *CNN-rand* is one of those CNN model variants that does not use word2vec, which has lower scores than other 3 word2vec-powered variants to some extent.

2.2 Sentiment Analysis for Turkish

A survey on Turkish sentiment analysis by Dehkharghani et al. (2016b) provides a thorough analysis of sentiment analysis, and propose a system of methods to analyze Turkish in this context. They propose an approach to process Turkish texts at different granularity levels. levels are proposed as *Word-level*, *Phrase-level*, *Sentence-level*, and *Document-level*. This work also maps several linguistic issues such as negation, intensification, conditional sentences, rhetorical questions, sarcastic phrases, and idiomatic uses. In addition, it also maps some of the other issues such as emoticons, conjunctions, domain-specific indicative keywords, and background knowledge. The work, however, does not provide solutions to all these issues. Some of them such as sarcastic phrases, and idiomatic uses, domain-specific indicative keywords, and background knowledge are left out as they need further research or language-specific tool-sets to extract related information regarding the related issues. The solution provided largely depend on SentiTurkNet (Dehkharghani et al., 2016a) framework, which is also developed by the same team.

One of the oldest researches on sentiment analysis in Turkish goes back to a thesis published by Erogul in 2009 where he applies a set of sentiment analysis processes to extract features from *Movie Reviews* dataset. In this work, he selects different sets of features based on frequencies, the root of words, part-of-speech and n-grams. He investigates performances of each method using different values.

In Vural et al. (2012), by using SentiStrength[1], Vural applied sentence-binary, sentence-max/min, and word-sum to acquire accuracy results on the same dataset. SentiStrength is a lexicon-based sentiment analysis library developed by Thelwall et al. (2010). This framework utilizes a pre-compiled list of positive/negative weights scored by humans for each word in different groups. Sentimental word list, booster word list, idiom list, negation word list, and finally emoticon list have different scores of their own. The framework then combines min and max scores within the text to determine a sentiment score for overall text. Vural is able to adapt the framework to Turkish and compile a result set for the same data set in Erogul (2009)'s work.

Firat Akba and Sever (2014) improved Vural et al. (2012)'s accuracy results on *Movie Reviews* significantly by employing Support Vector Machine(SVM) and Naive Bayes Classifier(NB) and also by including a feature selection step. They employed two main methods for feature selection stage. Results are presented by training an SVM and an NB classifier with features selected by both Chi-Square and Information Gain.

Demirtas and Pechenizkiy (2013) proposed and evaluated a model for sentiment analysis by machine-translating Turkish texts in *Product Reviews* data set. Demirtas and Pechenizkiy build this dataset for their this work. They both collect movie reviews from BeyazPerde[2] having the same size with benchmark English movie review dataset. They also collect smaller datasets from multidomain product reviews from a Turkish online retailer[3] website. In this work, it is aimed to translate datasets into English with machine learning(ML) and classify texts using different ML algorithms after translation. The benchmark dataset, which is originally in English, is also translated into Turkish and then back to English in order to measure effects of machine translation on accuracy results. Results are presented in three different ML algorithms; namely, Naive Bayes, Support Vector Machine, and Maximum Entropy classifications.

In this work, we both used *Movie Reviews* and *Product Reviews* data sets. Results acquired in these works are presented as our baseline scores in experiments section.

[1] http://sentistrength.wlv.ac.uk/
[2] http://www.beyazperde.com/
[3] http://www.hepsiburada.com/

CHAPTER 3

SENTIMENT ANALYSIS WITH VARIOUS SEGMENTATION METHODS AND DEEP LEARNING MODELS

In this thesis, it is aimed to investigate the effectiveness of various segmentation methods alongside well-known deep neural networks. Historically, Recurrent Neural Networks (RNNs) have been seen as the primary candidate for text classification (Socher et al., 2012, 2013; Turney and Pantel, 2010) due to various reasons. However, lately, further tests with Convolutional Neural Networks (CNNs) also provided at least as good results as RNNs. Both RNNs and CNNs have shown promising results (Yin et al., 2017) for English sentences. In this work, it is aimed to investigate and compare the performance of CNN and RNN for morphologically rich languages. In addition, we analyze the effect of segmentation method on the sentiment analysis performance for morphologically rich languages. We experimented with *4* major segmentation methods that yield vocabularies with different size and characteristics. We present the details of the segmentation methods, the CNN-rand (Kim, 2014) and the LSTM neural models in what follows.

3.1 Segmentation Methods

Segmentation helps divide each review into tokens. We primarily focused on *4* major approaches for segmenting raw text into tokens; namely, *Word-based*, *Sub-word*, *Morphology-based*, and *Hybrid* methods.

Word-based segmentation yields largest vocabulary; whereas, the vocabulary is the set of characters with the **character based** segmentation method. Besides, the vocabulary size, the length of input sequences increase as the token granularity is shifted from words to characters. In principle, *Word-based* model yields the largest vocabulary with shortest sequences. The trend is towards smaller vocabulary and longer sequences while moving from this point to *Hybrid*, *Morphology-based* and finally *Sub-word* respectively. We will provide more extensive statistics regarding vocabulary size and sequence length for each model applied in experiments.

We will be showcasing each segmentation method with following running examples.

Sentence 1: film bastan sona duygu somurusu ama anlayan nerde!

English: the movie *exploits*(typo) emotion from start to finish but who would care!

Sentence 2: geçen hafta elimize ulaştı, kullanımı kolay bulaşıkları pırıl pırıl yıkıyor.

English: we got it delivered last week, it's easy to use and it washes dishes very well.

3.1.1 ***Word-based*** Analysis

The *Word-based* approach is the most widely used model in recent studies. This is both due to its ease of use and its success being very reasonable for non-additive languages such as English. This is the case due to the fact that tokens in non-additive languages are mostly the same as word roots. However, this does not hold in case of additive languages such as Turkish. A root word could be converted to hundreds of distinct tokens just by using different suffixes. This might extend vocabulary to large numbers. Nevertheless, we also used this approach in this work to compare results to other approaches. In this work, we only deal with *Word-Token* model in our experiments.

Table 3.1: Running examples for *Word-based* segmentation methods

Segmentation	**Output**
Word-Token	film bastan sona duygu somurusu ama anlayan nerde !
Word-Token	geçen hafta elimize ulaştı , kullanımı kolay bulaşıkları pırıl pırıl yıkıyor .

For *Word-based* segmentation, *Word-Token* can be obtained by simply using the tokenized text directly. Table 3.1 shows *Word-Token* output for running examples.

3.1.2 ***Morphology-based*** Analysis

The *Morphology-based* analysis focuses on fragmenting each token into its building blocks by means of grammar rules for the language input text is written in. One challenge with the *Morphology-based* analysis is developing, finding and using morphologic tools in an efficient and correct manner. However, another challenge is to decide which information to keep and which to ignore. In this work, we used Zemberek (Ahmet Afsin Akin, 2007) for all morphological analysis related segmentations. We extracted embedded morphological information using several different approaches.

The full list of *Morphology-based* analysis variants is listed below:

- In *Lemma* approach, we extract lemma for each word. Rest of the word and suffixes are discarded.

Table 3.2: Running examples for *Morphology-based* segmentation methods

Segmentation	**Output**
Lemma	film bas So duygu somurusu âmâ anlamak Ner !
Lemma	geçen hafta el ulaşmak , kullanım kolay bulaşık pırıl pırıl yıkamak .
Lemma+Suffix/Meta	Noun A3sg Pnon Nom Noun A3sg Pnon Abl Noun A3sg P2sg Dat Noun A3sg Pnon Nom Unk Adj Adj PresPart Noun A3sg Pnon Loc Punc
Lemma+Suffix/Meta	Adj Adv Noun A3sg P1pl Dat Verb Pos Past A3sg Punc Noun A3sg Pnon Acc Adj Noun A3pl P3pl Nom Dup Dup Verb Pos Prog A3sg Punc
Lemma+Suffix	film A3sg Pnon Nom bas A3sg Pnon >dAn So A3sg In +yA duygu A3sg Pnon Nom somurusu âmâ anlamak +yAn Ner A3sg Pnon >dA !
Lemma+Suffix	geçen hafta el A3sg ImIz +yA ulaşmak Pos >dI A3sg , kullanım A3sg Pnon +yI kolay bulaşık lAr I Nom pırıl pırıl yıkamak Pos Iyor A3sg .
Stem	film bas so duygu somurusu ama anla ner !
Stem	geçen hafta el ulaş , kullanım kolay bulaşık pırıl pırıl yık .
Stem+Suffix/Meta	Noun A3sg Pnon Nom Noun A3sg Pnon Abl Noun A3sg P2sg Dat Noun A3sg Pnon Nom Unk Adj Adj PresPart Noun A3sg Pnon Loc Punc
Stem+Suffix/Meta	Adj Adv Noun A3sg P1pl Dat Verb Pos Past A3sg Punc Noun A3sg Pnon Acc Adj Noun A3pl P3pl Nom Dup Dup Verb Pos Prog A3sg Punc
Stem+Suffix	film A3sg Pnon Nom bas A3sg Pnon >dAn so A3sg In +yA duygu A3sg Pnon Nom somurusu ama anla +yAn ner A3sg Pnon >dA !
Stem+Suffix	geçen hafta el A3sg ImIz +yA ulaş Pos >dI A3sg , kullanım A3sg Pnon +yI kolay bulaşık lAr I Nom pırıl pırıl yık Pos Iyor A3sg .
Token-Meta	Noun Noun Noun Noun Unk Adj Adj Noun Punc
Token-Meta	Adj Adv Noun Verb Punc Noun Adj Noun Dup Dup Verb Punc

- In *Lemma+Suffix* approach, we also extract suffixes and concatenate them onto lemma as separate tokens. For suffixes lacking lexicon representation, the suffix class itself is used.

- In *Lemma+Suffix/Meta* approach, we extract lemma positional attribute. In addition to it, we also extract suffix classes for all suffixes. This model is a variant and we only added this to see whether word and suffix positional states hold significant information for sentimental content.

- In *Stem* approach, we extract stem for each word. Rest of the word and suffixes are discarded.
- In *Stem+Suffix* approach, we also extract suffixes and concatenate them onto stem as separate tokens. For suffixes lacking lexicon representation, the suffix class itself is used.
- In *Stem+Suffix/Meta* approach, we extract lemma positional attribute. In addition to it, we also extract suffix classes for all suffixes. This model is a variant and we only added this to see whether word and suffix positional states hold significant information for sentimental content.

It is important to note that in *Lemma* and *Stem* approaches, the negation is also discarded since the negation is a suffix. However, if the word root itself intrinsically holds negation, the meaning is not discarded.

For *Morphology-based* segmentations, we first extracted a dictionary containing all tokenized words used in datasets. Later on, a consumer processes the dictionary file to extract translation of each token into different representations for each of *Lemma*, *Lemma+Suffix*, *Lemma+Suffix/Meta*, *Stem*, *Stem+Suffix*, *Stem+Suffix/Meta*, *Token-Meta* models. Having this dictionary with the representation of each word integrated for each *Morphology-based* model, we can now traverse sentences in each review and encode tokens inside each sentence into corresponding token sets. Table 3.2 shows *Lemma*, *Lemma+Suffix*, *Lemma+Suffix/Meta*, *Stem*, *Stem+Suffix*, *Stem+Suffix/Meta*, *Token-Meta* outputs for running examples.

3.1.3 *Sub-word* Analysis

The *Sub-word* analysis is an approach to break down each word into its building blocks without thinking much about the meaning of the underlying morphology. By breaking down words, the vocabulary is shrunk by a considerable amount depending on the approach or the parameters used. In this section, we will deal with three major approaches. The first is Byte-pair encoding, which uses frequently used sub-word occurrences. The second one is n-gram, which we will be using 1-gram variant. Finally, the last one is syllable based segmentation.

Byte-pair Encoding (BPE) has been shown to be an effective way of dealing with large vocabularies in neural machine translation. Sennrich et al. (2015) propose to use sub-word units such as morphemes and phonemes for neural machine translation. These sub-word units are proposed to be extracted using BPE due to its robustness in automatically determining the morphemes and phonemes using the language corpora.

The BPE method first builds a vocabulary from a corpus iteratively by merging the frequently co-occurring token pairs. The number of iterations is defined in advance and determines the size of vocabulary. In a second step, segmentation is done by splitting the words into the tokens using the vocabulary built in the first step. The parts of words that can be reconstructed by the vocabulary tokens are retained as an individual token. The infrequent substrings in the text are broken down until they match a known toke. If not possible, they are discarded. This ensures that the vocabulary size of output text remains within desired limits. Note that, in BPE, there is no limitation on the length of vocabulary tokens and the tokens are not required to be meaningful on their own.

Since BPE requires a limit on vocabulary size for segmenting input text, we decided to work with different vocabulary sizes. 1000, 5000 and 3000 vocabulary sizes are used for *BPE-1k*, *BPE-5k*, *BPE-30k* approaches respectively.

N-gram Character Segmentation is an approach where text can be broken down into sub-word elements with a maximum size of n characters. For this work, we only used 1-gram, which can also automatically translated into character-based segmentation. It simply breaks downs words into single characters.

In Lee et al. (2016) authors used a character-based methodology where sentences are simply divided into characters. *Character* segmentation is a similar approach and basically a specific model of n-gram; namely, 1-gram. Each character will be a distinct token. The motivation behind this model largely comes from studies focusing on far-eastern languages such as Mandarin and Japanese. The alphabet for the majority of far eastern languages is made of self-identifying characters where each character has a distinct meaning. What's motivates us to use this model in case of Turkish is the fact that it is a member of Altaic language family. Therefore it might have deeply rooted morphological similarities to far-eastern languages.

The third sub-word text segmentation method stressed in this work is syllabification or hyphenation of text. Syllables can be automatically inferred from words in Turkish. (Aşlıyan and Günel, 2007) They are a core part of Turkish language and they are widely used (Çöltekin, 2014; Aşlıyan and Günel, 2007; Çoltekin et al., 2007) for Turkish NLP tasks. They also follow distinct patterns with a set of different forms due to deterministic nature of Turkish pronunciation, where each character almost exclusively represents the same sound or phoneme. Even though there are several different implementations of syllabification of Turkish texts, we decided to implement our own.

In order to implement an efficient way to encode words into their syllables we first identified syllable forms in Turkish. The regular syllable forms provided in Aşlıyan and Günel (2007) are as follows. We also identified a set of

Where V denotes vowels and C denotes consonants, regular and irregular syllable forms along with character sets are as follows.

Vowels (V): a, e, ı, i, o, ö, u, ü

Consonants (C): b, c, ç, d ,f, g ,ğ, h, j, k, l, m, n, p, r, s ,ş ,t ,v, y ,z

Regular Syllable Forms: V, VC, CV, CVC, VCC, CCV, CVCC, CCVC

Regular Syllable Examples: e, ev, ve, ver, erk, bre, mart

Irregular Syllable Forms: CC+V, CC+VC, VCC+, CVCC+, CC+VCC+

Irregular Syllable Examples: brre, trren, üfff, oturr, krrakkk (typos, foreign words or representation of sounds)

Along with syllable forms, we also wanted to identify word forms below. however, the list of possible word forms could be too large. Hence, we decided to only list cases where the order of syllable forms is not allowed.

V: V-CCVC (a-yran), CCVC-V (kral-a), CVCC-V (mart-ı), VC-V (at-a), CVC-V (ver-i), V-CCV (a-tkı)

CV: CV-CCVC (ka-lkan), CV-CCV (ko-lpa),

VC: CVCC-VC (türk-an), VC-VC (at-ak), VCC-VC (ary-an)

VCC: CVCC VCC

CCV: CCV CCVC (kra-ldan)

We decided to decode non-characters as distinct syllables. We also used a character normalizer which reflects special forms of vowels and consonants to their English counterpart. (i.e. [ı, İ, î, Î] –> i)

Table 3.3: Running examples for *Sub-word* segmentation methods

Segmentation	**Output**
BPE-1k	film ba st an sona du y g u s o m ur u su ama anla ya n nerde !
BPE-1k	geçen hafta el im iz e ul aş tı , ku ll anı mı kolay b ul aş ık ları p ır ıl p ır ıl yı kı yor .
BPE-5k	film bastan sona duygu so mu ru su ama anlayan nerde !
BPE-5k	geçen hafta eli miz e ulaştı , kullanımı kolay bul aşı kları pı rıl pı rıl yıkıyor .
BPE-30k	film bastan sona duygu so mur usu ama anlayan nerde !
BPE-30k	geçen hafta elimize ulaştı , kullanımı kolay bulaşıkları pırıl pırıl yıkıyor .
Character	f i l m b a s t a n s o n a d u y g u s o m u r u s u a m a a n l a y a n n e r d e !
Character	g e ç e n h a f t a e l i m i z e u l a ş t ı , k u l l a n ı m ı k o l a y b u l a ş ı k l a r ı p ı r ı l p ı r ı l y ı k ı y o r .
Syllable	film bas tan so na duy gu so mu ru su a ma an la yan ner de !
Syllable	ge çen haf ta e li mi ze u laş tı , kul la nı mı ko lay bu la şık la rı pı rıl pı rıl yı kı yor .

For *Sub-word* segmentations, we used three different methods.

- *Character* model can be easily extracted by splitting each token inside each review into its building characters. Each character is then used as separate tokens.
- For *BPE-1k*, *BPE-5k*, *BPE-30k* models, an encoder developed by Stuart Axelbrooke[1] is used. In order to use this encoder, it be should first fit into dataset corpus. Therefore, we first extracted a corpus by compiling entire content from all datasets. Since we are going to use different vocabulary sizes for BPE segmentations, we created 3 separate encoders with vocabulary sizes of $1k$, $5k$, and $30k$ respectively. After encoders are trained with the corpus, they are ready to be used. BPE encoders both encode the words and returns identifications for each token extracted from input sentence. Therefore, for BPE segmentations no further identification method will be applied.
- For *Syllable* segmentation method we implemented and used publicly available python-syllable[2] package. The package is able to tokenize, transform, inverse-transform input into and out of syllables. It is also able to apply basic filters based on syllable frequencies extracted from a publicly available Turkish news[3] dataset. The package is also able to extract new syllable vectors by fitting new datasets.

Table 3.3 shows *Character*, *BPE-1k*, *BPE-5k*, *BPE-30k*, *Syllable* outputs for running examples.

3.1.4 ***Hybrid*** Analysis

Dual Decomposition (Wang et al., 2014) models segmentation as an optimization problem for selecting whole-words or underlying characters as base tokens. This model is proposed to overcome the challenges faced by Chinese. We adopt and modify this approach as a ***Word-Character Hybrid*** model for Turkish such that we use characters-based segmentation of unknown words. The words which are not recognized by Zemberek (Ahmet Afsin Akin, 2007) are being broken down to its characters while the rest is kept as whole words.

Table 3.4: Running examples for *Hybrid* segmentation methods

Segmentation	**Output**
Hybrid	film bastan sona duygu s o m u r u s u ama anlayan nerde !
Hybrid	geçen hafta elimize ulaştı , kullanımı kolay bulaşıkları pırıl pırıl yıkıyor .

For *Hybrid* segmentation, the dictionary created for *Morphology-based* analysis is reused. Representation provided by *Token-Meta* hold information about the type

[1] `github.com/soaxelbrooke/python-bpe`
[2] github.com/ftkurt/python-syllable
[3] kaggle.com/ahmetax/hury-dataset

of the token. When the token is not recognized by our morphological analyzer *Zemberek*, *Token-Meta* will have the value of *Unk*. Since for *Hybrid* model, dual decomposition requires using *Character* based segmentation and *Word-Token* for known words, *Hybrid* is simply extracted by compiling these two by also checking the value of *Token-Meta* for each token. Table 3.4 shows *Hybrid* output for running examples.

3.2 Deep Neural Networks

For classification job, we decided to use deep neural networks instead of well-known machine learning such as Support Vector Machine (SVM) or Naive Bayes (NB) classifiers. This is both due to earlier works extensively using these classifiers and the fact that there is no work that actually uses Deep Learning models for classification in Turkish.

CNNs and RNNs are very well known for text classification jobs. Therefore we decided to utilize both of them in our work. For CNN we decided to use a special variant CNN-rand proposed by Kim (2014). For RNN, we decided to use a special form of RNN which is called Long-Short Term Memory (LSTM) network.

For our investigation for efficient text classification approaches be complete, we should also show the extent of deep neural networks' usage on text classification task. In part 2 we already listed some prominent earlier work on both text classification in general and on Turkish sentiment analysis. The earlier work shows that there is a wide area of research and on topic and various different approaches in this regard. In pre-processing part, we already went through a wide range of segmentation methods. In this part, we will also focus on different deep neural network methods used for the task. We will focus on a special implementation of Convolutional Neural Network CNN-rand (Kim, 2014) and also another widely used special Deep Neural Network implementation LSTM.

3.2.1 Convolutional Neural Networks

Convolutional Neural networks or ConvNets are Deep Learning models designed to minimize the need for data pre-processing. They use a set of different multilayer perceptrons to achieve various types of non-linearity for modeling input layers to the next stages. CNNs are also known to be shift invariant, which means they tend to be behaving the same wherever the focal point for the occurrence of a feature of interest is. Figure 3.1 shows an illustration for convolutional neural networks.

3.2.2 CNN-rand

We used one of CNN model variations described in Kim (2014) for our neural network. (See figure 3.2) This variation initiates embedding with random values, while the rest depend on an external word2vec. CNN-rand does not use word2vec; hence, it learns the embedding from scratch. We opted this model variation to minimize our dependence on a language corpus, in addition to the fact that it is frequently reported that the

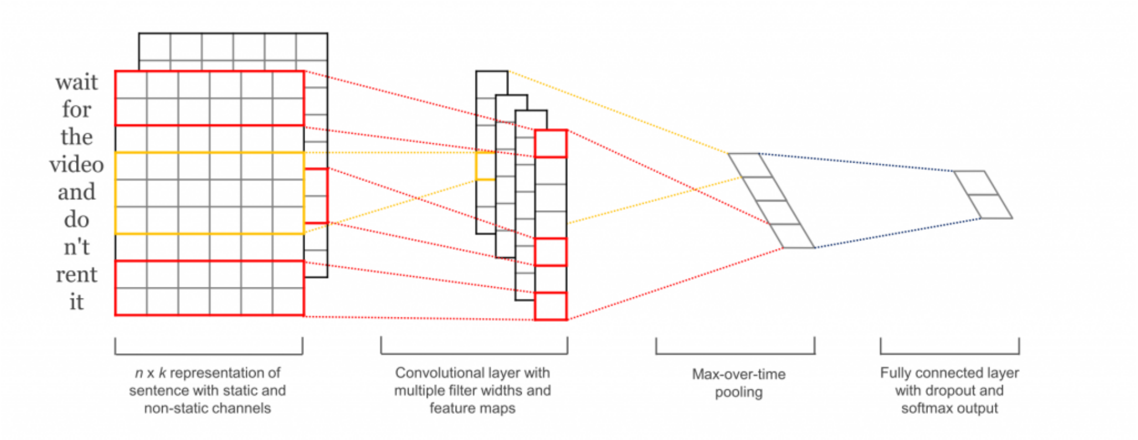

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 3.1: Figure showing a potential utilization Convolutional Neural Networks for text classification job (Kim, 2014)

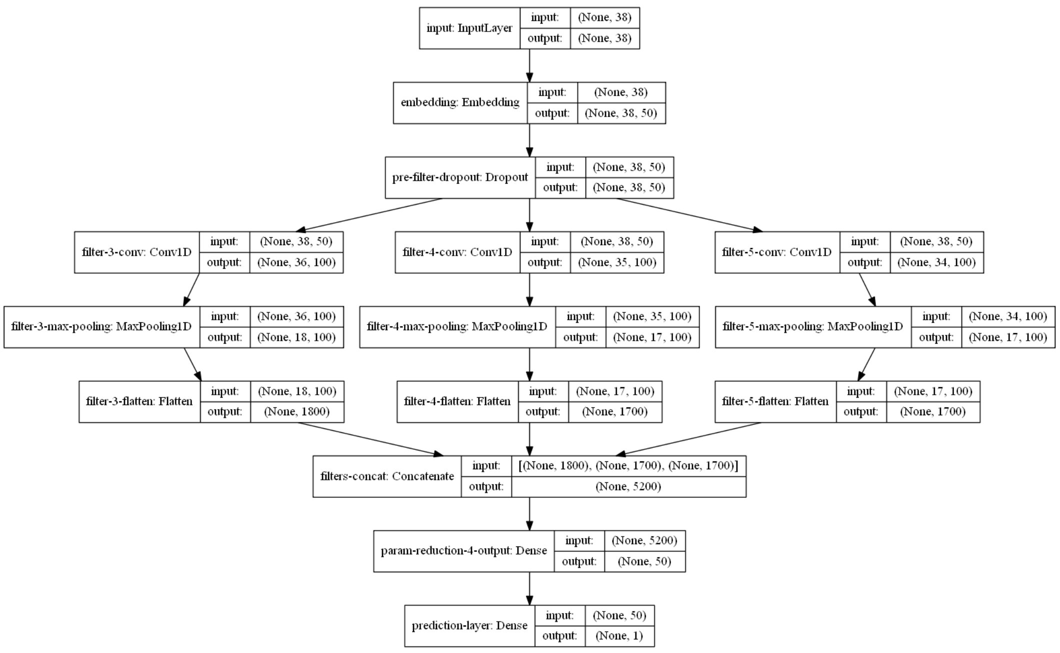


Figure 3.2: CNN-rand network described in Kim (2014) with layer shapes produced for *Word-Token* segmentation on Movie Reviews dataset

initial embeddings contribute very little to the network success in earlier research. We extended source code cnn-text-classification-tf[4] implementation.

In this approach, data is first converted to a digital representation by extracting numerical identification for each token. To fit resulting output to a vectorial space of equal sized batches, each review is then padded with special <PAD> tokens until it matches the longest entry.

[4] `github.com/dennybritz/cnn-text-classification-tf`

Key layers introduced in this model (Figure 3.2) are described below.

Input Layer: This layer receives the input data both during training and evaluation. This layer has an input shape of (None, 38) which indicates that every input sequence introduced has to have a length of 38, which is the maximum length for movie reviews dataset when processed with *Word-Token* segmentation. The *None* part indicates the batch size flexibility, which can be any number.

Embedding Layer: This layer keeps the vector representation of each token, in this case, words. The embedding size is introduced as 50, therefore the input shape attached by a depth of 50 at the output of the layer, which represents the vector size for tokens.

Dropout Layer: Dropout layer is a layer which helps network learn seemingly insignificant patterns within data by randomly dropping a proportion of connections to the next layer. A dropout value of 0.5 is used for this layer. Therefore, it means that every time a data is introduced 50% of connections from upper Embedding layer will be discarded and will not be passed onto lower layers.

Filtering Layers: This layer groups introduces a system filtering mechanisms to look deeper into phrases and groups of words inside input sequence. Paths are provided for filter sizes of 3, 4, and 5. Each path represents a filtering mechanism and has following sub-layers.

Convolution Layer: This is the main convolution layer which introduces CNN part of the neural network. Convolutions are basically a smaller sized patch convolving over an input layer and passing the measured weights onto next layers. Next layers usually have smaller width and length and larger depth. Therefore, the convolutional layer translates spatial information into depth, which could be considered as a temporal output. For each of convolution layers in these filters, the patch sizes are selected as the size of filters. Therefore, the convolutions will be looking for filter-sized frames or in other words phrases. This setting helps to extract sentiment which is held by a group of words as opposed to the singular entries. Since the stride size is the filter size, the output layer is smaller by a factor of $stride - 1$, i.e. an input size of 38 is converted to the output size of 36 for filter size 3.

Max Pooling Layer: This layer translates the input layer into smaller representations using Max pooling. In this case, the output is half the size of the input layer, which means each output parameter is obtained from the maximum value of corresponding 2 input parameters.

Flatten Layer: This layer converts multi-dimensional parameter setting into single dimension. This is needed for concatenation layer at the end of filtering stages. In case of filter size of 3, an input layer of (None, 18, 100) is converted to (None, 1800), effectively reshaping input while keeping all parameters.

Concatenation Layer: This layer concatenates inputs from different layers and output them in a single shape. The output shape is simply a shape with a summation of input layers with no parameter being lost. In this case, input shapes of 1800, 1700, and 1700 are converted into an output shape of 5200.

Dense Layer: This layer gradually reduces the number of parameters in order to achieve the prediction size. Sigmoid activation is used for this dense layer. In this case, the layer converts an input size of 5200 into an output of size 50.

Dense Layer for Prediction: This is the second dense layer which translates an input size of 50 into a single parameter which is needed for a binary sentiment analysis task as in our case.

Key parameters for CNN models indicated by Kim (2014) are *word embedding dimension*, *dropout*, *filter size* and a *number of filters*.

The *word embedding* represents the vectorial space for each vocabulary within input text. Dropout is a quite radical idea implemented in deep neural networks to let network better understand building blocks of the meaning the network is trying to extract from the input. *Dropout* is the ratio of connections between two layers being randomly dropped during training. This lets network learn lower weight connections explaining input. *Filter sizes* indicate the number of tokens to convolve over during training, and filters variable defines a number of filters for each filter size.

3.2.3 CNN-rand Simplified

Simplified CNN-rand (Figure 3.3) is a smaller network version of the original CNN-rand network. It eliminates some of the filters and reduces sizes to a great extent. The simplification is proposed by alexander-rakhlin[5] due to several listed reasons.

The changes introduced in this simplified version of CNN-rand are listed below.

- *Smaller embedding dimension is used*: 20 instead of 300
- *Fewer filter size is used*: 2 filter sizes instead of 3
- *Fewer filters are used for each size. Proposes experiments showing that fewer is enough*: 3-10 filter size instead of 100
- *Proposes random embedding initialization is no worse than word2vec init on IMDB corpus*: CNN-rand is preferred (as we already did)
- *Network slides over Max Pooling instead of original Global Pooling.*

Key layer changes introduced in this model (Figure 3.3) are described below.

Input Layer: This layer is kept the same as the original.

Embedding Layer: The embedding size for vector representations is reduced to 20 in this model.

Dropout Layer: This layer is kept the same as the original.

[5] `github.com/alexander-rakhlin/CNN-for-Sentence-Classification-in-Keras`

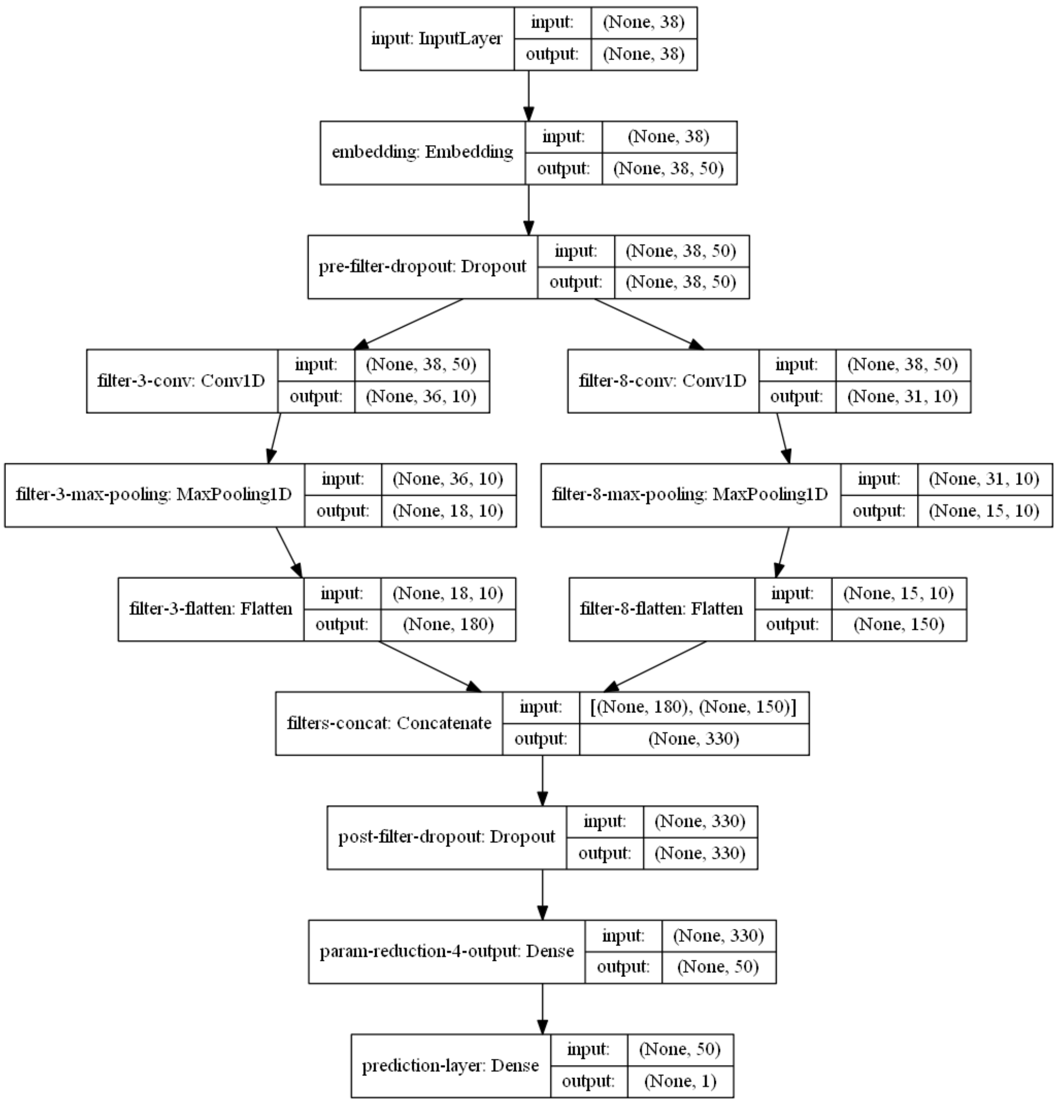


Figure 3.3: Simplified CNN-rand network proposed by alexander-rakhlin with layer shapes produced for *Word-Token* segmentation on Movie Reviews dataset

Filtering Layers: Filter sizes are changed in this model. Filter sizes of 3, and 8 are used instead of the original sizes of 3, 4, and 5. Sub-layers for filters are used as in the original model.

Concatenation Layer: This layer is kept the same as the original.

Dropout Layer: A new dropout layer is introduced with a dropout value of 0.8 between concatenation layer and the dense later.

Dense Layer: This layer is kept the same as the original.

Dense Layer for Prediction: This layer is kept the same as the original.

We will run our experiments on this network alongside the original CNN-rand. We will validate if above changes and justification hold the truth about being harmless to the performance.

3.2.4 Long-Short Term Memory (Neural Network)

Long-Short Term Memory network is a special type of Recurrent Neural Network. In this RNN variant, a memory cell is incorporated into RNN cell which accumulates information throughout input propagation. This ensures that network can recognize input patterns over long intervals. LSTM was first proposed by Gers et al. (1999). The model was later investigated and improved by Hochreiter and Schmidhuber (1997).

An LSTM cell consists of 4 main components; a cell, an input gate, an output gate and a forget gate. The mathematical formulation of these components is detailed as follows.

Let $x_t \in R^d$ (input vector to the LSTM block), $f_t \in R^h$ (forget gate's activation vector), $i_t \in R^h$ (input gate's activation vector), $o_t \in R^h$ (output gate's activation vector), $h_t \in R^h$ (output vector of the LSTM block), $c_t \in R^h$ (cell state vector), and $W \in R^{h \times d}$, $U \in R^{h \times h}$ and $b \in R^h$ (weight matrices and bias vector parameters which need to be learned during training), following formulas (Graves et al., 2013) determine the runtime activity of an LSTM cell:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{3.1}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{3.2}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{3.3}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{3.4}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{3.5}$$

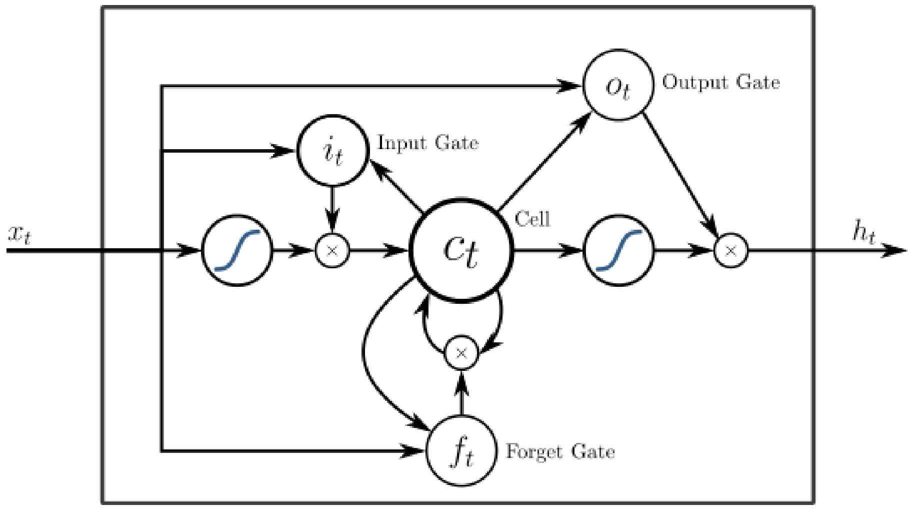


Figure 3.4: An LSTM cell detailed in Graves et al. (2013) with input, output, and forget gates.

3.2.5 Proposed LSTM Model

Long-Short Term Memory (LSTM) cells are widely used in time series. In most cases, spoken language is seen as a similar problem to the time series problem, since the order of spoken words shows similarities to the time series data since both are sequential and temporal as opposed to the image visual analysis. LSTM cell is specifically designed for temporal data processing. Each cell holds information about earlier tokens inside an entry and outputs a value based on current value and earlier tokens.

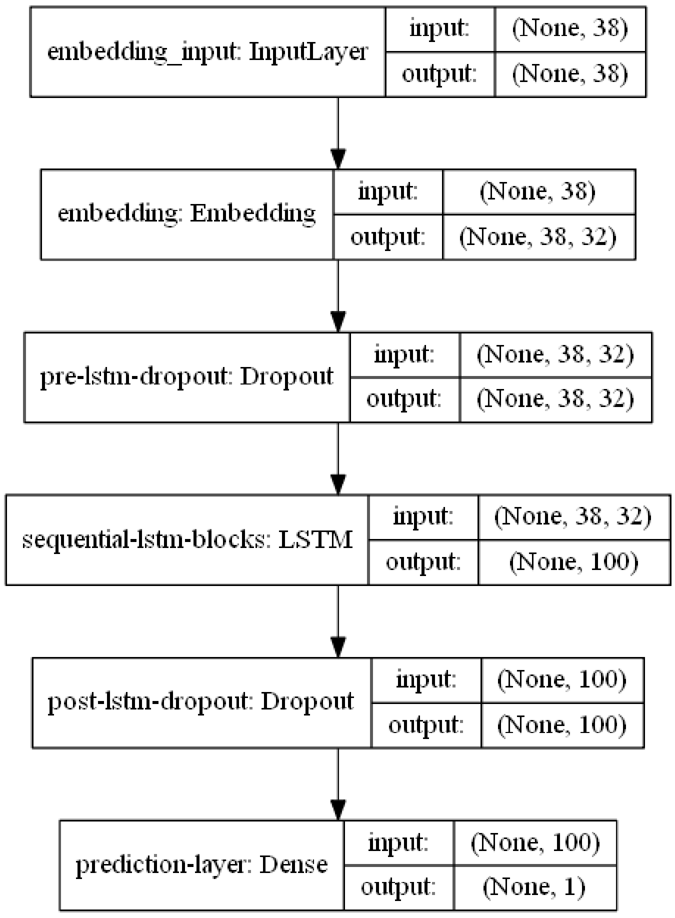


Figure 3.5: A simple LSTM network with layer shapes produced for *Word-Token* segmentation on Movie Reviews dataset

Key layers introduced in this model (Figure 3.5) are described below.

Input Layer: This layer receives the input data both during training and evaluation. The layer attributes are the same as attributes described for CNN models.

Embedding Layer: For this layer, the embedding size of 32 is introduced, therefore the input shape is attached by a depth of 32 at the output of the layer, which represents the vector size for tokens.

Dropout Layer: A dropout value of 0.2 is used for this layer. Therefore, it means that every time a data is introduced 20% of connections from upper Embedding layer will be discarded and will not be passed onto lower layers.

LSTM Blocks: This layer introduces LSTM blocks for sequence processing. Further details for LSTMs is provided in section 3.2.4. This layer helps to extract sentiment using memory, processing, forgetting mechanisms built into the LSTM cells.

Dropout Layer 2: Another dropout layer is introduced after LSTM blocks for this model, and a dropout value of 0.2 is used.

Dense Layer for Prediction: This layer translates an input size of 100 into a single parameter which is needed for a binary sentiment analysis task as in our case.

We also tested our datasets with a neural network utilizing LTSM cell blocks. We extended our code from MachineLearningMastery[6]. Figure 3.5 shows the network components.

A very important feature of LSTM networks is their capability to process data with multiple layers. This enables to process data in different ways to extract various data in various forms, and process all extracted data in different layers. One possible application would be using different segmentation methods as different layers. However, this work aims to compare performances of different segmentation methods. Therefore, we decided not to use multiple layers for our LSTM network. On a side note, using multiple layers could also make a final comparison between CNN and LSTM results a bit unfair as CNN network that we use does not support multiple layers.

3.2.6 Hyper-parameter Optimizations

Deep learning hyper-parameters are neural network controls that provide fine tuning capabilities. They can be used to fine-tune a model in order to produce best results for different goals and datasets.

Zhang and Wallace (2015) indicate that they already used a grid search for hyper-parameter tuning and that selected parameters are identified as being the most effective. In the article, it is also mentioned that L2 has little effect on results. We decided to run a grid search for hyper-parameters to validate the effectiveness of the chosen parameters in Zhang and Wallace (2015) on the face of changing dataset and segmentation methods.

To test the effectiveness of the selected parameters and also to validate earlier remarks about L2 regularization, we decided to run a grid search. The grid-search is executed with following hyper-parameters and corresponding value sets.

Filter Sizes: (3,4,5), (10,16,22), (16,22,27), (22,27,33)

Dropout: 0.4, 0.5, 0.6

L2 Regularization: 0.0, 0.001, 0.01, 0.1

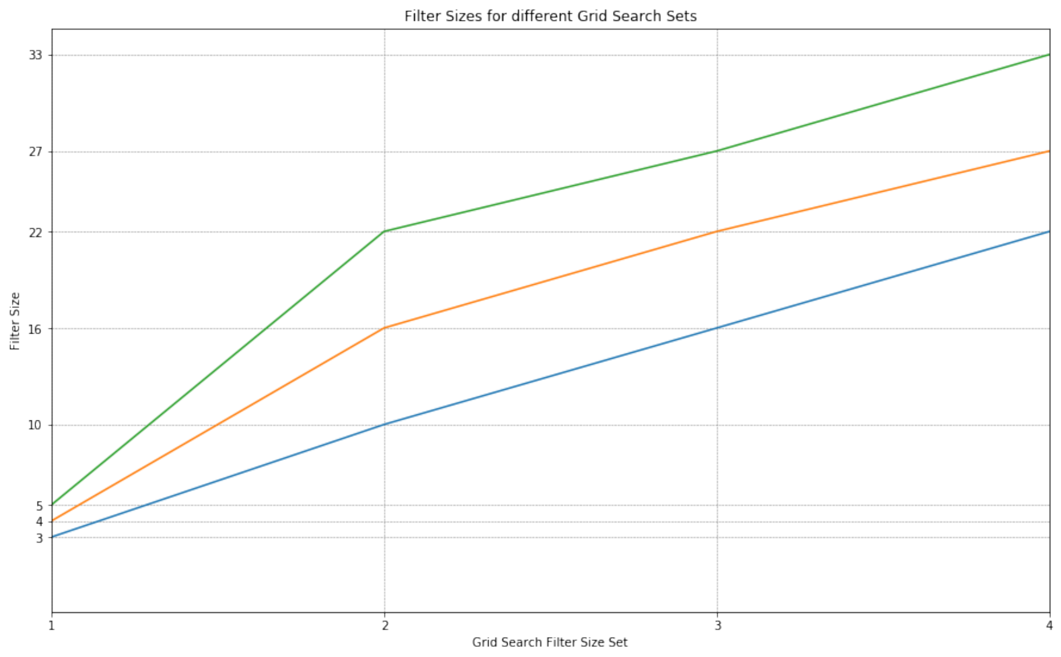


Figure 3.6: Filter sizes used for Grid Search

Filter sizes are determined using character number statistics on our combined datasets. The number of tokens per each word is highest for *Character* segmentation method. Therefore, we decided to use statistics with this segmentation method. The average number of characters in words is 5.49. Therefore, we decided to use a linear space between original filter sizes (3,4,5) to around 5.5 times these values. However, after first filter size set, we also limited the difference between concurrent filter sizes to make sure the model does not lose any information with phrases with the length in between. Figure 3.6 shows the final distribution of filter sizes for all sets used for grid search.

We trained a CNN-rand model with *Character* segmentation output for book reviews dataset with varying values of hyper-parameters provided earlier. Filter sizes used in test case are derived from the average number of characters in each word within the dataset. We thought this selection criterion for filter sizes would be convenient because the core justification for filters is provided in the original paper such that they will encompass varying number or words within each sentence in order to capture the sentiment held by these groups of preceding words. However, our pre-processing methods divide each tokenized word into multiple numbers of tokens depending on the segmentation method provided. The reason for *Character* selection for the job is the fact that this divide is the most significant in *Character* based segmentation.

Figure 3.7 indicates that 0.5 dropout value and 3,4,5 filter size set, which is the default parameter set for original work, derive the best results. It also shows that L2 adds no positive effect to the performance. As a result, for the experiments, we set dropout parameter to 0.5 and filter sizes to $3, 4, 5$. We did not use any L2 regularization and left it at the default value of 0. We also used recommended default levels for other

[6] `machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/`

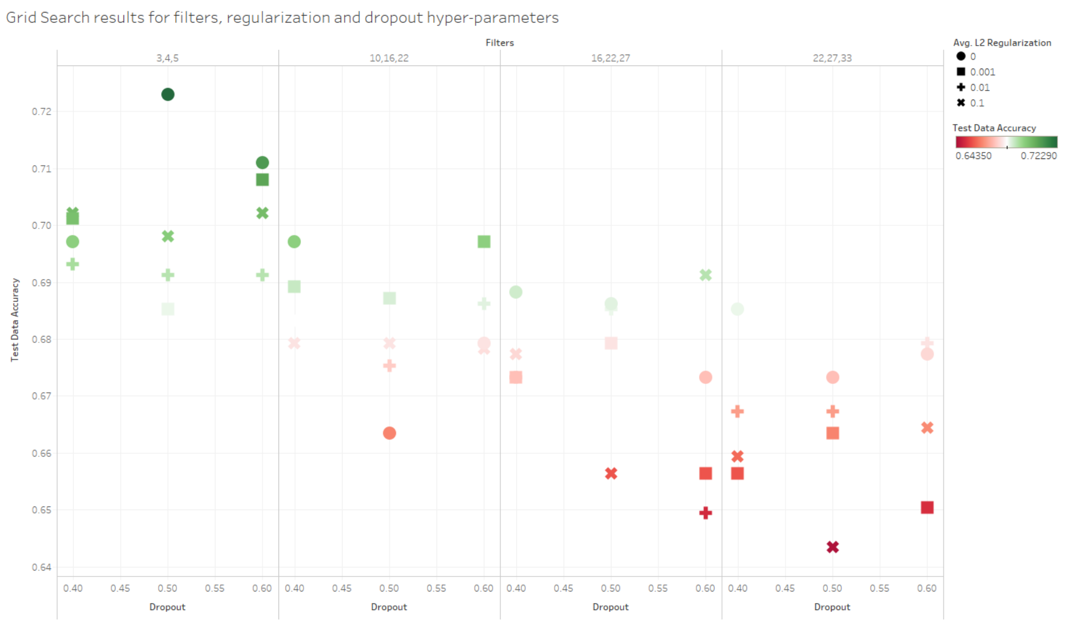


Figure 3.7: Accuracy graph for Dropout broken down by Filters. Color shows Accuracy. Shape shows details about L2 Regularization.

parameters.

3.2.7 Model Training Callbacks

For deep learning model building, we used Keras[7] with a Tensorflow[8] back-end. In order to build a neural network, the training data is presented to the model repetitively. This lets model better grasp training data and learn new things after model decides to do something different during an earlier repetition. This is a common practice and each iteration is called an *Epoch*. At the end of each epoch, deep learning framework (Tensorflow) and the interface (Keras) provide access to run a set of functions before deciding if it is desired to continue with more epochs. These functions are called *Callbacks*.

There are also common practices in terms of functions called during between epochs. One of the definitive Callbacks is running the model on Validation data and measuring Loss, Accuracy and etc. Other frequently used callbacks are BestModelSave, EarlyStopping, LearningRateDecay and progress plotters.

During our model building, we used following callbacks.

BestModelSave: This callback tracks some parameters and saves model when the best value so far is encountered. The variables to be tracked in order to save model is indicated when the callback is created.

[7] `github.com/fchollet/keras`
[8] `github.com/tensorflow/tensorflow`

EarlyStopping: When model training starts a constant epoch number should be provided. However, we cannot know for sure how many epochs will be sufficient for the model to learn training data. Therefore, as a rule of thumb, we provide a relatively high number of Epochs in order to make sure we trained the model sufficiently. However, if not interrupted, the model will most definitely over-fit to the data. Over-fitting is as bad as under-fitting; therefore, it is generally a good idea to stop training before that happens. Early stopping tracks variables of choice to stop training when configured monitors indicate that over-fitting started. *EarlyStopping* needs monitoring parameter, minimum delta, and patience. Callback monitors the monitoring variable and constantly expects this parameter getting better. When it does not get better beyond minimum delta value it deducts from patience value. If the patience value reaches 0, training is interrupted. Every time a better result for monitoring variable is encountered, the patience value is reset.

Progress Monitor: These monitors mostly plot progress with training, epochs, accuracy, loss and expected remaining time.

We used *Validation Accuracy* as *BestModelSave* monitor. Therefore, whenever the model achieves the highest accuracy on Validation data so far this callback saves the model.

For all models, we used validation loss as *EarlyStopping* monitoring parameter, and 0.001 as minimum delta. Due to underlying differences between LSTM and CNN networks, we used 2, and 20 as patience value respectively. LSTM networks need few epochs (1-3) to fully train model. We used 200, and 5 as the number of Epochs for CNN, and LSTM networks respectively.

We developed our own *Progress Monitor* for tracking training of our models. It plots the progress of accuracy and loss for validation and training data. It also indicates the position of *BestModelSave* and *EarlyStopping* patience count-down. It also saves progress visualization at the end of training cycle.

3.3 Limitations

This work aims to demonstrate effective ways of dealing with sentiment extraction task for informal Turkish text with the least possible interference on input data. Therefore, this work does not utilize advanced Natural Language Processing (NLP) tools. Hence, operations listed below are not covered within this work, and they are proposed as an extension to this work in the conclusion and future work (Chapter 5) section.

Typo Checker: This work does not perform typographical error checking and correction. The work aims to analyze the performance of different segmentation methods on informal texts. Utilizing such tools could have clouded our results from fully understanding the effects of studied segmentation methods and utilized neural networks.

Normalization: Datasets include a fair amount of information encoded in different representations such as numbers, emoticons, shapes, images and etc. Even though handling these issues are shown to contribute to the results positively, we decided to use the raw form of input texts and propose this as future work.

Disambiguation: Another shortcoming of morphological analysis demonstrated in this work is its inability to extract the correct form of each word and use the accurate roots and suffixes. However, this requires a deep understanding of language structures and grammar. It also requires the implementation of such advanced tools. This work does not rely on availability of such advanced tools, using them would increase the processing complexity.

Sentence Modelling: It is also possible to model sentence structures and use this information to train more effective models for sentiment analysis. However, this will also add another layer of complexity to this work.

Global Word Vectors: In this work, a set of different segmentation methods are utilized, and experiments for these methods are conducted on different datasets. Therefore, following shortcomings can be emphasized for this decision.

- Global word vectors defines vectors for whole words, but this work focuses on breaking down words into sub-word elements by various segmentation methods. The resulting tokens will not have any representation within these ready to use vector definitions. The available ones are very likely to be pointing to the incorrect vectors. For example, in a case where *kamu-flaj* (camouflage) is broken down to *kamu* (public) + *flaj* (no meaning) the resulting vector for *kamu* will point out to the inaccurate vector representation.
- Being a morphologically rich language creates scarcity for a very large number of rarely used word-suffix sets in Turkish. Hence, scarce tokens even if available within the WordVec library might have inaccurate representation due to lack of suitable sample text to train the vectors in a correct way.
- The baseline scores indicate results achieved by running respective experiments on cited datasets. Using a global corpus for all datasets or training model using data form all datasets will decrease accuracy of the models, since it will be forcing model to learn much more than the original baseline experiments. This will lead to an unfair comparison between results in this work and the baseline scores.

CHAPTER 4

EXPERIMENTS

4.1 Data Sets

In the experiments, we used two benchmark datasets. *Movie Reviews* is a collection of movie reviews from the Turkish movie platform Beyazperde[1]. The dataset contains 54K annotated paragraph length reviews originally used in Erogul (2009). The dataset is collected from Turkish movie review site Beyazperde from various movies at random. Beyazperde users can rate movies on the scale of [0.5-5]. The dataset entries are labeled as follows. Reviews rated as 4.0-5.0 are accepted as positive, 2.5-3.5 as neutral and 0.5-2.0 as negative reviews. *Movie Reviews* data set size info is as shown in Table 4.1.

Table 4.1: *Movie Reviews* initial statistics

Dataset	Negative	Positive
Reviews	27.000	27.000

The *Product Reviews* first used by Demirtas and Pechenizkiy (2013) is compiled from online retailer websites. It consists of 5 different datasets. In Demirtas and Pechenizkiy (2013) authors aim to achieve plausible results in text classification task for Turkish text by using machine translation. Therefore they collect reviews from BeyazPerde by the same amount as their benchmark IMDB review dataset, which is in English. In addition to this movie reviews dataset, they also compile smaller datasets by collecting reviews for various products from online retailer Hepsiburada.com. The reviews are compiled for four different product categories, namely electronics, DVDs, kitchen products, and books in addition to the movie review dataset. For simplicity, we will be calling the full set of these 5 datasets as *Product Reviews.*

During dataset compilation, each review is labeled based on the rating user provided for the product upon review. Since the majority of the votes are 3+ on a scale from 1 to 5, classification is exerted as $1-3$ rates being negative and $4-5$ being positive. *Product Reviews* dataset size info is as shown in Table 4.2.

In text classification task, particularly when working with machine learning methodologies, vocabulary size and data geometry are very important. Therefore, we also obtained basic statistics on datasets. Vocabulary size, average sentence length and

[1] `www.beyazperde.com`

Table 4.2: *Product Reviews* initial statistics

Dataset	Negative	Positive
Movie	5.331	5.331
Book	700	700
Electronics	700	700
Kitchen	700	700
DVD	700	700

maximum word count per review is provided in Table 4.3 for both *Movie Reviews* and *Product Reviews*.

Table 4.3: Vocabulary sizes and other basic statistics for *Movie Reviews* and *Product Reviews* datasets.

	Avg. Sentence Length	Max Review Size	Vocabulary
beyaz_perde.neg	3.76	801	108,682
beyaz_perde.pos	3.73	1,717	112,566
book.neg	3.69	235	7,367
book.pos	3.51	126	5,634
dvd.neg	3.41	293	7,553
dvd.pos	3.30	289	6,564
electronics.neg	3.91	281	7,791
electronics.pos	3.58	245	5,904
kitchen.neg	3.58	130	6,475
kitchen.pos	3.44	210	5,457
movie.neg	2.36	59	19,481
movie.pos	2.41	59	17,825

On the other hand, the distribution of review length in terms of words seem to be varying to a great extent for various datasets. Figure 4.1 shows differences in the distribution of sentences among different datasets.

4.2 Data Pre-processing

In this part, we will thoroughly discuss data preparation phases and the justifications for each process. These processes compile into our preprocessing stage. In data pre-processing, we are first breaking each review into its sentence components using sentence detection functions provided by Zemberek, (Ahmet Afsin Akin, 2007) and sentences are tagged as their parent(review the sentence is extracted from) reviews. In the second phase, each sentence is processed with different segmentation methods discussed in chapter 3. The main motivation behind this is the fact that neural networks are known to perform better with single sentence entries, due to the fact that

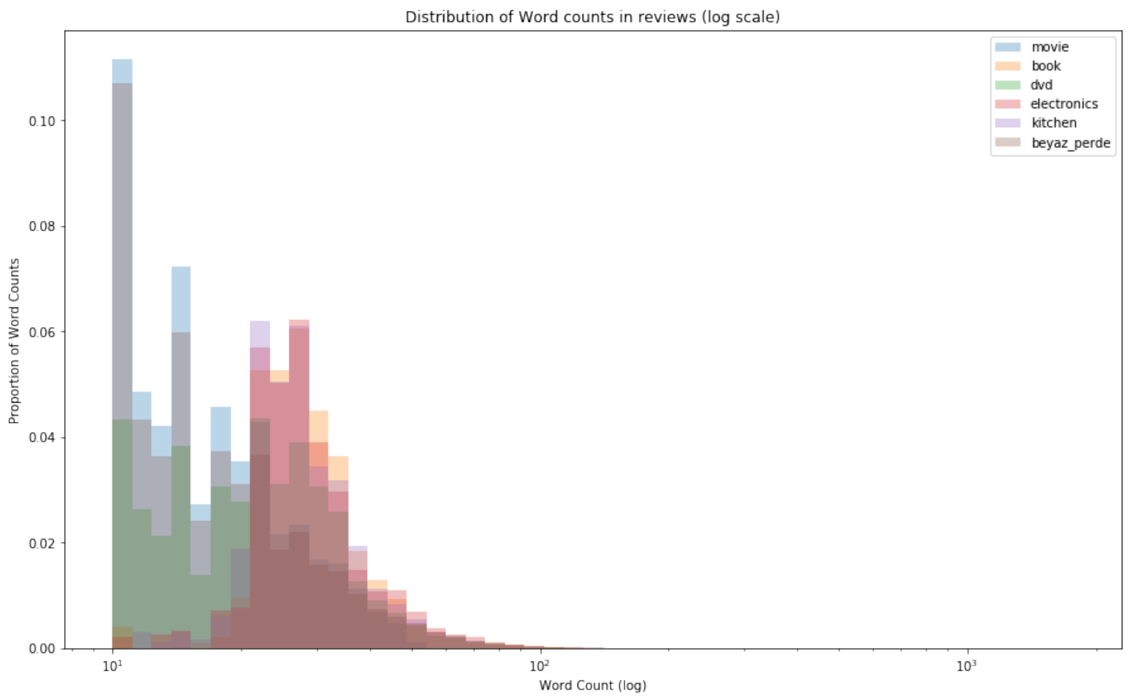


Figure 4.1: Histogram showing the distribution of review length for datasets.

a sentence mostly contains one distinct sentiment. Whereas, multiple sentences, even if they are in the same review, could contain different sentiments. Use of sentence, in this regard, is a common practice for Deep Learning model training for sentiment analysis in English.

In this work, we are planning to use Deep Neural Network models for extracting sentiment analysis. DNN models work better and more efficiently with smaller input shapes and single-sentence inputs. However, our datasets consist of reviews, most of which constitutes several sentences. In order to tackle this problem, we are planning to model our reviews into sentences and smaller parts. In this part, we will also discuss approaches and processes involved in the sentence-level review modeling.

The third stage is identification phase. In this stage, each token from segmentation method output for each dataset is identified and an ID is assigned. This phase is needed due to neural networks need numerical IDs rather than textual entries. The deep learning models provide identification methods, but we stuck to our custom identifiers for two reasons. The first is that some Turkish characters might be misidentified with those methods, the other is that we also wanted to identify each token with their respective frequency. Our identifiers will assign smaller ID numbers for more frequent tokens. This ensures that a smaller dataset will lack high-valued IDs and thereby have a smaller vocabulary size.

In the last stage, we split datasets into *Test*, *Validation*, and *Training* subsets.

4.2.1 Review Sentence Break-down

Most of the methodologies discussed in related work and our baseline researches use machine learning models such as Support Vector Machine and Naive Bayes. These models work pretty well with smaller datasets. However, deep learning methods work better with larger datasets. In fact, when the dataset is smaller than intended, the network is at high risk of over-fitting. Over-fitting happens when network memorizes the dataset itself and abandons understanding underlying patterns inside the dataset. When larger datasets are provided, the network is forced to learn these patterns. Datasets in *Product Reviews* with the exception of *movie reviews* have 700 negative and 700 positive reviews for each. A 1400-reviews-large dataset is very small to train a neural network. Not to mention, a substantial amount of reviews in these datasets will be used for validation and test splits. The original work also mentions this shortcoming and they decide to enhance dataset by machine-translating an English dataset with similar product reviews. (Demirtas and Pechenizkiy, 2013)

On the other hand, the neural networks we decided to use and the majority of others accept training data only when each entry has the same size. This means that each review which will be fed into the neural network will have the size of review with the maximum size. Reviews smaller than the largest review in terms of lengths/number of tokens will be padded with special PAD tokens.

When input dimensions are large, the convolution layers will be larger in a sense that there will be a need convolve over a wider length. Therefore, the network will consume much higher memory and computational power. The network will also need more training to learn connections for all new input cells appended to convolution layers.

It is also important to note that LSTMs are known to perform poorly with longer sentences due to their inability to compensate the distance between tokens and establish states corresponding to relationships between far-apart words in order to extract the sentiment. These issues are discussed in Raffel and Ellis (2015), and in response an *Attention* layer is proposed and employed by many types of research in various fields. (Larochelle and Hinton, 2010; Bahdanau et al., 2014; Luong et al., 2015)

Table 4.4: *Movie Reviews* and *Product Reviews* post-split stats

Dataset	Negative	Positive
Beyaz Perde	101.643	100.667
Book	2.582	2.455
Movie	12.572	12.837
Electronics	2.739	2.504
Kitchen	2.507	2.406
DVD	2.387	2.309

To eliminate all these problems, we decided to break reviews into sentences. And we will also label each sentence as parent review. Sentence-based dataset statistics are as shown in Table 4.4. As seen in the table, the dataset sizes after review-split are significantly larger than the original 1400. With this process, we have datasets with minimum sizes around $5k$ entries.

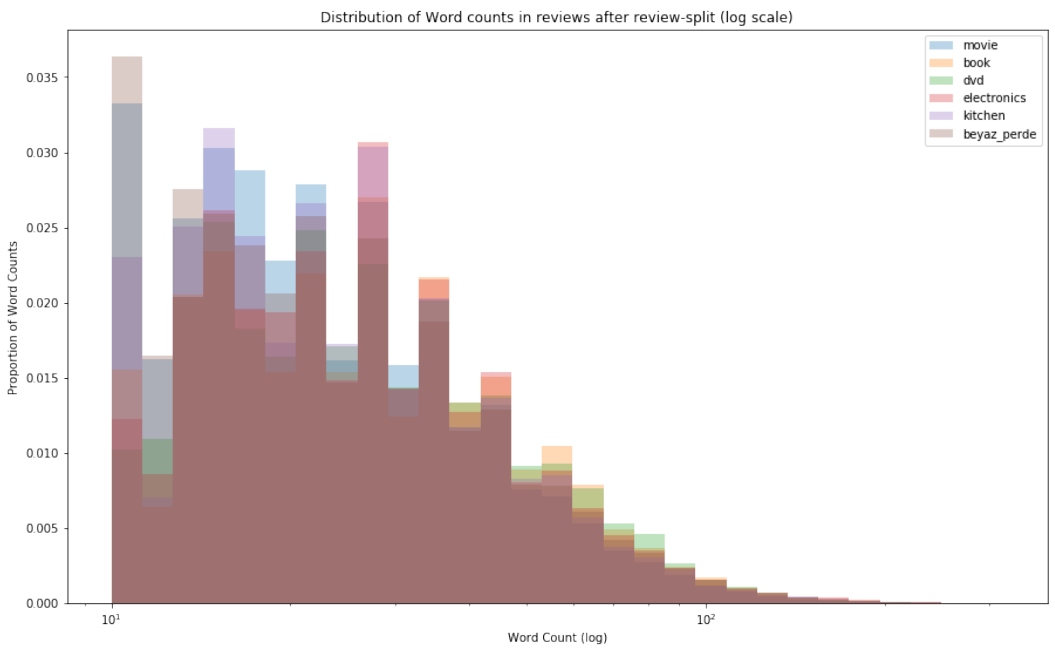


Figure 4.2: Histogram showing the distribution of review length for datasets after review-sentence-split.

Figure 4.1 shows the differences in the distribution of sentences among different datasets after sentence-level review split. This figure shows that our sentence length distribution is much more whole than the initial distributions.

However, we can still see some datasets with too long entries. By manual inspection of the dataset, we noticed that some reviews are too long with few or no sentence stops. They mostly skip punctuation altogether. This ensures that our issue with long reviews persists even after sentence-level review split.

4.2.2 Forced Review Break-down

In order to address the issue with too long sentences, we decided to forcefully break these entries down into smaller entries. We first investigated the distribution of sentence sizes. We noticed that sentences with larger word counts tend to be a very small minority. Therefore, we decided that we can easily solve the issue without causing much damage. We decided to apply a filter with $percentile = 99.5$. This will ensure that only sentences within longest 0.5% of entries will be broken down.

Table 4.5 shows new maximum size and old values for each dataset. We can see that, all datasets are now in similar geometries.

Table 4.5: *Movie Reviews* and *Product Reviews* maximum sentence size after forced split

	New Size	Old Size
Beyaz Perde	55	309
Book	44	94
Movie	38	51
Electronics	54	121
Kitchen	48	99
DVD	48	91

4.2.3 Sentiment Loss with Review Break-down and Majority Voting

In this work, we are investigating the efficient ways to classify a sentence which might have a negative or positive sentiment. In doing so, we assume that each review intrinsically has a sentiment communicated by the author. However, the sentiment will not be possibly grouped into binary clusters in which a review will have an absolute positive or negative sentiment. The author might have simply chosen to communicate a sentiment which is in between. In this case, we will have several sentences some of which might contain negative sentiment while other hold positive sentiment at the same time. However, when we break-down reviews into sentences and blindly label them as their parent reviews, we are introducing a bias which will affect our results. In order to compensate the loss of sentiment at review level, we decided to apply a post-processing stage which will evaluate accuracies at review level by incorporating results for each sentence from each review into a single score by the means of *Majority Voting*.

In order to achieve majority voting, in a correct manner, we must also take ownership of sentences while splitting data into *Test*, *Validation*, and *Training* subsets. We must ensure that all sentences from a single review end up in the same split. Since the test data is the split which will determine accuracy results, this is particularly important for *Test* split.

4.2.4 Segmentation

In segmentation stage, we created global corpus by compiling all dataset contents. This corpus is then used to create, build and use various encoders for different segmentation methods. These segmentation methods and involved processes are detailed in section 3.1.

4.2.5 Token Identification

Token identification is a stage where we enumerate different tokens in each segmentation model. The identifier will distribute identities assigning a respective count number to each token based on their position within overall dataset or corpus. However, the identities distributed at this stage are temporary. During this first stage frequency of all tokens are also calculated. At the end of this process, each token will have a temporary ID and the calculated frequency. Using these frequencies, new and permanent identities are generated by sorting tokens in descending frequency values.

Table 4.6: Compiled Vocabulary sizes and Id counts for each segmentation methods

Segmentation	**Vocabulary / ID Count**
BPE-1k	957
BPE-30k	28.676
BPE-5k	4.822
Character	112
Hybrid	117.294
Lemma	99.872
Lemma+Suffix/Meta	96
Lemma+Suffix	99.972
Stem	99.766
Stem+Suffix/Meta	96
Stem+Suffix	99.863
Word-Token	198.262
Token-Meta	15

During the identification, identities are generated for the aggregated corpus, and therefore will not depend on a single dataset. However, the identification corpus will be compiled for each segmentation method separately. Knowing this, sorting identities by descending token frequencies, we ensure that a smaller dataset will most probably have tokens with lowest identity numbers. This will affect network size in a positive way.

Table 4.6 shows vocabulary sizes for each segmentation method. However, it seems the vocabulary size is still exceptionally large for some of the segmentation methods. This is particularly true in case of *Word-Token* which has a vocabulary size of 198*k*.

In order to overcome this large vocabulary problem and possibly reduce the vocabulary size further, we did further investigation. Figure 4.3 shows that 60% of all word-tokens occurs only once within the entire text. Around 80% of them occur less than 3 times. Scarce words are very problematic in training an efficient model. Having too many of these will inflate the network embedding. On the other hand, network will not able to learn anything significant from them due to their scarcity. On the contrary, the weights could be allocated in a way that they can determine the sentiment of the sentence in a very wrong way when those words occur. Therefore, we decided to remove these scarce words by using a frequency limit of 3. Hence, a word that occurs less than 3 times does not have the chance to be in Train, Validation and Test data samples at the same time.

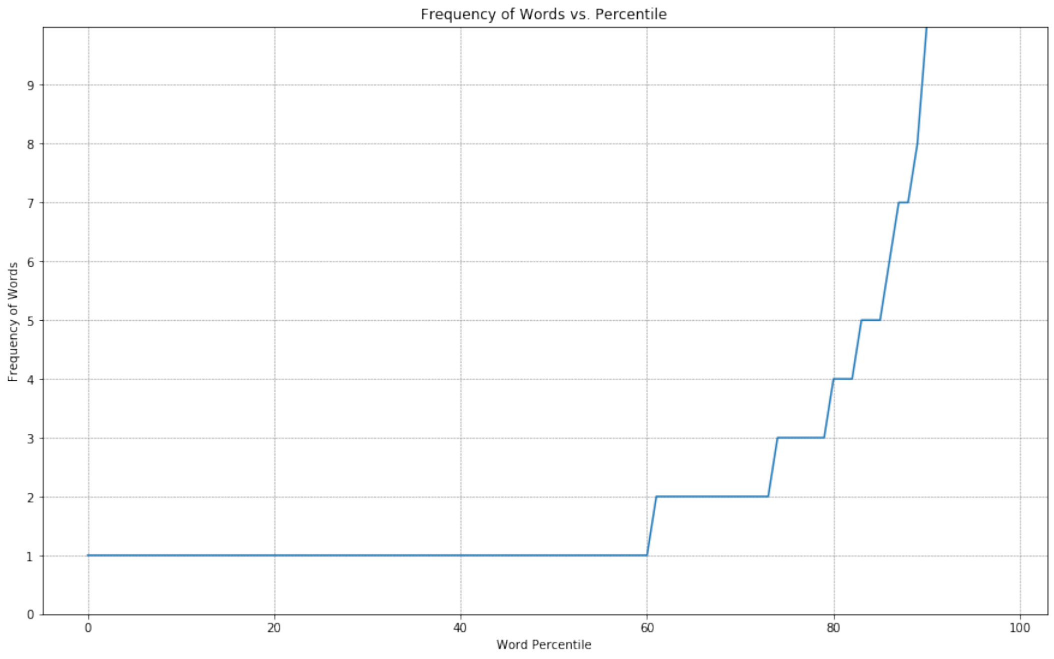


Figure 4.3: Word-token frequency distribution over percentile. Y-axis is limited to 10 in order to show lower frequencies better.

4.2.6 Sentence-Level Review Modeling for Train-Test Split

Machine learning models need a training stage where they can extract a pattern from a dataset in order to use them in future for unsupervised classification. A common practice is dividing a labeled dataset into test and training samples. In this way, model training and testing can be done with different samples. This will ensure that the score calculated for the model will be based on data that network never encountered so far. This is crucial in determining the true success of model and also to see if the model is over-fitted to the training data.

First of all, in case of lack of this split, the network will be already familiar with the data and will tend to know better what to do with the test data. The model will have far more optimistic results than the actual performance. Secondly, performance analysis will be completely blind to over-fitting problems. With familiar data, no matter how far over-fitting goes, the results will get better and not worse.

To overcome these challenges, training-test data split is introduced. However, in our work, we went to one step further to include also a validation sample.

The justification for validation data is not much different from that for the training-test split; however, this time it is due to different reasons. During network building and hyper-parameter tuning, some information is secretly moved from validation data into the model itself over time. This happens through multiple iterations on model building and testing. After every iteration, we tend to change some of the parameters in order to achieve better scores on validation data. This also means we tune the model to get better results with validation data, which mean information intrinsic to the validation data is moved over to the model.

Using separate samples for validation and testing ensures that no information is moved from Test set into the model building stage. Thereby, it makes final performance analysis much more reliable.

Apart from the problem of splitting data into three separate samples, we also introduced another complexity parameter into the process during sentence-level review break-down Part 4.2.1. The Majority Voting process adds a requirement such that each sentence should be traceable back to the parent review. Creating an interface to enable tracking each sentence to its parent review will enable us to calculate Majority Voting scores in the post-processing stage. However, we should also use this interface while splitting data into training, validation and test samples. We need to do this split by keeping sentences from the same reviews in the same sample.

Under these circumstances, in order to split data accurately, we first identified each review in each dataset and assigned a unique ID to each. Datasets are then split into sentences while holding sentences with their parent label and parent identity. Train, Validation, and Test split is performed at review level rather than sentence level. Sentences inside each review are then extracted for each sample in order to compile the actual sample.

This process introduces two consequences,

- The number of reviews in each sample will be equal to the training/validation/test ratio rather than the number of sentences, which was originally intended.
- The number of sentences in each sample might deviate from the intended ratio due to different reviews having a different number of sentences.

These two items will cause no problem, as long as the eventual ratio of these samples does not deviate much from the intended ratio.

Table 4.7: Number of sentences in each sample after training/validation/test dataset split.

Dataset	Train	Test	Validation
Beyaz Perde	162.561	20.393	20.467
Book	4.001	515	548
Movie	20.473	2.530	2.520
Electronics	4.278	466	528
Kitchen	3.961	493	485
DVD	3.789	463	468

During experiments, we used review-level training/validation/test split with ratios of 0.8, 0.1, and 0.1 respectively. Table 4.7 shows the distribution of sentences into samples for each dataset. The table shows that the ratios of training/validation/test data do not deviate significantly from original ratios intended. Therefore, we don't expect any problem in terms of asymmetry.

4.3 Deep Neural Networks

For our sentiment extraction job, we used a set of different Deep Neural Network tools. The primary Deep Learning candidates for the job were Convolutional and Recurrent Neural Networks. For CNNs, we used CNN-rand and Simplified CNN-rand detailed in subsection 3.2.2 and subsection 3.2.3. For RNNs, we used Long-Short Term Memory Neural Network detailed in subsection 3.2.5. We also a set of model training monitors detailed in subsection 3.2.7 in order to increase efficiency and accuracy of the trained model.

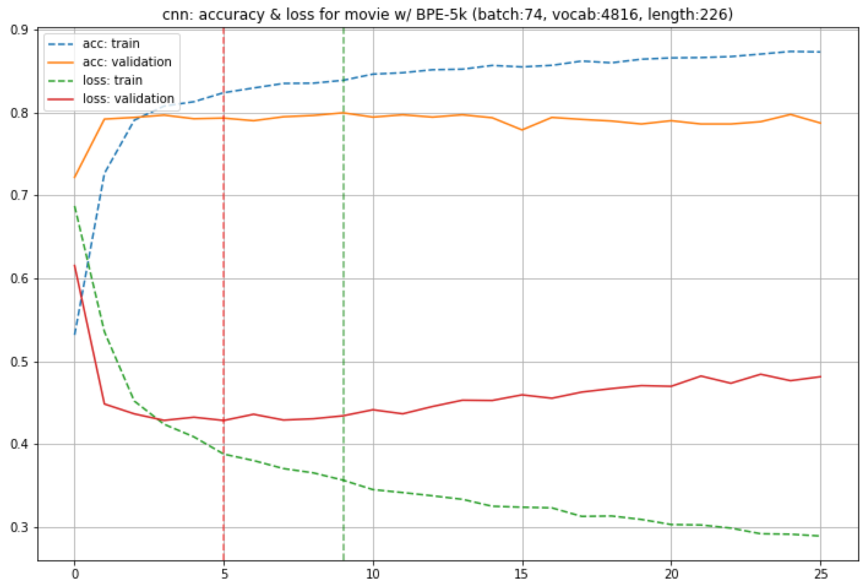


Figure 4.4: CNN-rand: Progress of accuracy and loss for validation and training data on Movie dataset w/ *BPE-5k* segmentation. The green vertical line shows the position of best model save, while orange one shows patience count-down for EarlyStopping.

Figure 4.4 shows output created by our custom progress monitor detailed in subsection 3.2.7. The figure shows that model is able to learn new things until 5th epoch where the minimum loss value of validation data is achieved. *EarlyStopping* counts down patience from this point on. 20 epochs later, at the end of Epoch 25, the training is interrupted due to lack of progress in monitoring variable, Validation Loss. Note that the loss is deteriorating while training data loss and accuracy gets better. This means that model is over-fitting. On the other hand, the vertical dashed green line shows the position of the best model when the highest accuracy on validation data is achieved. Model is saved on disc at this point. After training ends at epoch 25, the best model is reloaded from the disc in order to predict test data.

4.4 Baseline Methods

In this work, we used baseline scores for *Movie Reviews* and *Product Reviews* datasets from earlier text classification researches for which these datasets were used. For *Movie Reviews* data set we have 5 different scores from 2 different types of research which belong to Vural et al. (2012); Firat Akba and Sever (2014). For *Product Reviews*, we have 4 different scores from the same research that belongs to Demirtas and Pechenizkiy (2013).

Movie Reviews data set is first compiled by Erogul (2009) for his thesis where he applied a set of sentiment analysis processes to extract features from the data set. In this work, he selects different sets of features based on frequencies, the root of words, part-of-speech and n-grams. He investigates performances of each method using different values.

Later on, Vural et al. (2012) uses SentiStrength[2] on the same dataset. He applies sentence-binary, sentence-max/min and word-sum to acquire accuracy results on *Movie Reviews* data set. The results acquired by Vural et al. (2012) by adapting the framework to Turkish is also listed in our baseline scores for *Movie Reviews* data set.

Table 4.8: Baseline Scores on the *Movie Reviews* data set

Model	Author	**Acc.**
Sentence-binary	Vural et al. (2012)	70.39
Sentence-max/min	Vural et al. (2012)	74.83
Word-sum	Vural et al. (2012)	75.90
Chi-Square	Firat Akba and Sever (2014)	**83.90**
Information Gain	Firat Akba and Sever (2014)	**83.90**

Firat Akba and Sever (2014) improves Vural et al. (2012)'s accuracy results on *Movie Reviews* by employing Support Vector Machine(SVM) and Naive Bayes Classifier(NB) and also by including a feature selection step. They employ two main methods for feature selection stage. Results they acquire by training an SVM and an NB classifier with features selected by both Chi-Square and Information Gain are also listed in our baseline scores for *Movie Reviews* data set.

Baseline scores for *Movie Reviews* dataset are listed in Table 4.8.

For *Product Reviews* dataset, scores from Demirtas and Pechenizkiy (2013) are used as baseline scores. In their work, it is aimed to translate datasets into English with machine learning(ML) and classify texts using different ML algorithms after translation. They have results for three different ML algorithms; namely, Naive Bayes, Support Vector Machine, and Maximum Entropy classifications. Table 4.9 shows their scores on these datasets.

We also looked into choices made for training and test data splits for literature work

[2] `http://sentistrength.wlv.ac.uk/`

Table 4.9: Baseline Scores on the *Product Reviews* data set

Model	**Movie**	**Book**	**DVD**	**Electronics**	**Kitchen**
Naive Bayes[1]	69.50	**72.40**	**76.00**	**73.00**	**75.90**
Naive Bayes MT[1]	**70.00**	71.70	74.90	64.40	69.60
Linear SVC[1]	66.00	66.60	70.30	72.40	70.00
Linear SVC MT[1]	66.50	66.90	67.60	64.40	67.30

[1]Demirtas and Pechenizkiy (2013)

experimenting on our chosen datasets. However, none of the works mentioned (Vural et al., 2012; Firat Akba and Sever, 2014; Demirtas and Pechenizkiy, 2013) reports this information. A common approach is keeping test and validation sets minimal while keeping training data large enough to train the neural network. Therefore we decided to use 10% for testing and validation sets. The remaining 80% is used for training. More info is provided in part ch:exp:review-modeling. The resulting set sizes for different datasets are also provided in table 4.7.

4.5 Experiment Results and Discussion

Segmentation methods presented in Part 3 gives the chance to execute a lot of experiments. During experiments, we built data output for 13 different segmentation methods for 6 different datasets. We finally executed classification modeling on 3 different deep learning models. In total, we executed CPU and GPU intensive 234 deep learning model building jobs.

In this part, we will present our results for CNN-rand, Simplified CNN-rand, and LSTM. However, because of reasons we listed in part 4.2.1, raw results will not reflect the true performances of segmentation methods and Deep Learning models. This is because some sentences inside a review may not have the same sentiment as the overall review. In such a case, even if the classifier classifies the sentence accurately, we will count it as wrong due to its parent review having the opposite label. To make our results comparable we should calculate review-level accuracy. Therefore, we created a post-processing step which will also calculate Majority Voting results. In each section, we will present both raw results and scores after Majority Voting.

In this results section, we will present our scores for both *Movie Reviews*, and *Product Reviews* in the same table. Our core segmentation methodologies (see list below) will be listed at the top in each results table. Other methodologies will be listed at the bottom.

Base models and others mentioned above are as follows.

- *Base Models*: *Word-Token*, *Character*, *BPE-1k*, *BPE-5k*, *BPE-30k*, *Syllable*, *Hybrid*, *Lemma*, *Lemma+Suffix*, *Stem*, and *Stem+Suffix*

- *Other Models*: *Lemma+Suffix/Meta*, *Stem+Suffix/Meta*, and *Token-Meta*

The best score for each dataset will be highlighted on the table. Tables will also have a calculated column for average scores for segmentation methods over all datasets. The maximum average score will also be highlighted.

4.5.1 CNN-rand Results

Table 4.10: CNN-rand Accuracy Results w/o Majority Voting

	Movie	Book	DVD	Electr.	Kitchen	Beyaz Perde	Avg
Word-Token	82.29	68.54	65.23	**74.03**	68.36	**74.07**	**72.09**
Hybrid	82.17	68.35	64.15	73.39	66.73	73.31	71.35
Lemma	81.78	67.77	62.42	72.75	**70.18**	72.72	71.27
Lemma+Suffix	83.16	69.32	64.36	69.10	67.95	73.60	71.25
BPE-5k	82.06	66.41	64.79	73.39	66.73	72.57	70.99
Syllable	77.83	69.32	62.63	72.96	69.57	72.43	70.79
Stem	82.29	**69.51**	63.28	68.88	67.75	72.76	70.75
BPE-30k	**83.32**	69.32	61.77	72.75	61.05	74.02	70.37
BPE-1k	78.46	65.05	**68.03**	66.09	60.45	71.82	68.32
Stem+Suffix	82.65	68.74	64.15	46.78	68.97	73.18	67.41
Character	74.70	65.05	57.24	63.73	60.85	68.49	65.01
Lemma+Suffix/Meta	60.40	60.39	57.88	60.94	58.01	59.32	59.49
Stem+Suffix/Meta	62.65	55.15	60.04	59.66	58.62	59.49	59.27
Token-Meta	58.62	57.28	55.51	54.94	52.33	54.31	55.50

We present our raw CNN-rand results in Table 4.10. This table shows accuracy results for sentence-level classification. Score table shows that *Word-Token* holds the highest average score of 72.09%. It is followed by *Hybrid*, *Lemma*, and *Lemma+Suffix* with similar scores respectively.

We also present our majority voting CNN-rand results in Table 4.11. This table shows accuracy results for review-level classification. Score table shows that *Word-Token* holds the highest average score of 80.81%. It is followed by *BPE-5k*, and *Hybrid* with similar scores respectively.

4.5.2 Simplified CNN-rand Results

We present our raw CNN-rand results in Table 4.12. This table shows accuracy results for sentence-level classification. Score table shows that *Lemma+Suffix* holds the highest average score of 72.65%. It is followed by *BPE-5k*, *BPE-30k*, and *Stem* with similar scores respectively.

We also present our majority voting CNN-rand results in Table 4.13. This table shows

Table 4.11: CNN-rand Accuracy Results w/ Majority Voting

	Movie	Book	DVD	Electr.	Kitchen	Beyaz Perde	Avg
Word-Token	90.52	76.43	74.29	**79.29**	76.43	**87.91**	**80.81**
BPE-5k	90.14	80.00	73.57	75.71	**78.57**	86.11	80.68
Hybrid	**90.80**	77.14	72.86	77.86	76.43	86.91	80.33
Lemma	89.11	75.00	72.14	78.57	78.57	85.98	79.90
Lemma+Suffix	90.61	77.86	**75.71**	71.43	72.14	87.19	79.16
Syllable	86.10	**80.71**	68.57	75.71	77.14	86.17	79.07
BPE-30k	90.42	76.43	74.29	75.00	68.57	87.81	78.75
Stem	89.30	79.29	70.00	65.00	75.00	86.70	77.55
BPE-1k	88.26	77.86	73.57	71.43	67.14	86.37	77.44
Stem+Suffix	90.14	75.71	**75.71**	45.71	74.29	86.76	74.72
Character	81.41	73.57	62.14	68.57	67.86	82.24	72.63
Lemma+Suffix/Meta	66.20	67.14	62.14	67.86	62.14	67.85	65.56
Stem+Suffix/Meta	66.01	47.86	63.57	62.14	58.57	67.76	60.99
Token-Meta	62.72	59.29	55.71	57.86	48.57	58.04	57.03

Table 4.12: Simplified CNN-rand Accuracy Results w/o Majority Voting

	Movie	Book	DVD	Electr.	Kitchen	Beyaz Perde	Avg
Lemma+Suffix	82.85	**72.62**	65.44	72.10	69.78	73.08	**72.65**
BPE-5k	**83.04**	68.54	**66.31**	71.89	**70.99**	72.80	72.26
BPE-30k	82.92	69.13	66.09	**74.46**	66.73	**73.72**	72.18
Stem	81.62	69.51	63.93	72.53	67.95	71.72	71.21
Lemma	82.29	68.54	64.58	70.39	68.97	71.76	71.09
Stem+Suffix	82.57	70.29	62.42	72.10	66.33	72.80	71.09
Hybrid	79.49	67.57	63.93	72.75	67.14	72.97	70.64
Word-Token	82.92	69.51	62.20	67.81	66.94	73.44	70.47
Syllable	80.36	67.57	60.91	72.75	68.15	70.79	70.09
BPE-1k	78.81	68.54	62.85	69.31	67.55	70.50	69.59
Character	73.95	64.27	57.24	65.67	58.82	65.93	64.31
Stem+Suffix/Meta	60.95	63.69	56.59	61.16	63.08	58.24	60.62
Lemma+Suffix/Meta	61.15	63.88	57.45	58.37	58.82	58.16	59.64
Token-Meta	55.85	57.67	55.29	53.43	48.68	53.82	54.12

accuracy results for review-level classification. Score table shows that *Lemma+Suffix* holds the highest average score of 81.27%. It is followed by *BPE-5k*, *Hybrid*, and *Stem* with similar scores respectively.

When we compare results for CNN-rand and Simplified CNN-rand neural network models, we can see that there is no significant difference between performances of each. In fact, Simplified CNN-rand scores a bit higher than original CNN-rand imple-

Table 4.13: Simplified CNN-rand Accuracy Results w/ Majority Voting

	Movie	Book	DVD	Electr.	Kitchen	Beyaz Perde	Avg
Lemma+Suffix	90.61	**81.43**	76.43	77.86	75.00	86.30	**81.27**
BPE-5k	90.61	75.71	76.43	75.00	**80.00**	86.46	80.70
Hybrid	89.01	75.71	75.00	**80.71**	77.14	85.96	80.59
Stem	88.92	**81.43**	76.43	77.14	73.57	84.67	80.36
Word-Token	**91.08**	77.14	75.00	73.57	77.14	86.81	80.13
BPE-30k	89.95	73.57	**77.14**	76.43	76.43	**87.04**	80.09
Stem+Suffix	88.83	77.86	74.29	80.00	72.14	85.83	79.82
Syllable	87.79	79.29	70.71	77.86	78.57	84.67	79.81
Lemma	88.92	76.43	74.29	73.57	73.57	85.09	78.64
BPE-1k	89.11	75.00	71.43	76.43	72.86	84.17	78.16
Character	83.76	69.29	60.71	71.43	62.14	79.78	71.18
Lemma+Suffix/Meta	67.23	72.14	64.29	57.86	59.29	66.87	64.61
Stem+Suffix/Meta	66.20	72.14	61.43	54.29	62.14	65.74	63.66
Token-Meta	58.69	59.29	58.57	50.00	49.29	57.83	55.61

mentation. Average of all scores measured by the original CNN-rand implementation is 74.61%, while the same average is 75.33% for Simplified CNN-rand implementation.

In the light of this information, we will completely discard scores acquired by the Original CNN-rand implementation. In following parts and chapters, we will only take scores measured for Simplified CNN-rand and we will simply refer Simplified CNN-rand scores as CNN-rand scores.

4.5.3 LSTM Results

We present our raw CNN-rand results in Table 4.14. This table shows accuracy results for sentence-level classification. Score table shows that *Word-Token* holds the highest average score of 72.05%. It is followed by *Stem*, *Lemma*, *BPE-5k*, and *Lemma+Suffix* with similar scores respectively.

We also present our majority voting CNN-rand results in Table 4.15. This table shows accuracy results for review-level classification. Score table shows that *Word-Token* holds the highest average score of 81.36%. It is followed by *BPE-5k*, and *Lemma* with similar scores respectively.

4.5.4 Results in Comparison to Baseline Scores

In this section, we will compare our results with baseline scores. We will present comparisons of *Movie Reviews* and *Product Reviews* separately. Since we have too many segmentation methods and multiple numbers of classifiers, we will only present our results with highest average scores for each classifier. In order to stay consis-

Table 4.14: LSTM Accuracy Results w/o Majority Voting

	Movie	Book	DVD	Electr.	Kitchen	Beyaz Perde	Avg
Word-Token	**83.16**	67.57	**67.60**	**73.39**	66.73	73.86	**72.05**
Stem	81.54	69.51	66.09	72.75	69.37	72.63	71.98
Lemma	82.17	69.71	65.66	69.74	69.98	72.97	71.70
BPE-5k	80.95	67.77	**67.60**	72.96	66.13	73.93	71.56
Stem+Suffix	82.25	69.51	64.36	71.24	**67.34**	73.54	71.38
BPE-30k	82.81	67.57	66.09	68.24	66.94	**74.44**	71.01
Lemma+Suffix	82.25	**69.90**	66.52	70.39	62.27	73.81	70.86
Hybrid	79.17	67.57	61.12	71.46	66.53	72.78	69.77
Syllable	78.93	69.51	60.69	71.89	65.52	71.97	69.75
BPE-1k	76.68	66.21	64.36	70.17	62.07	70.99	68.41
Character	65.42	60.00	55.72	58.37	55.38	67.13	60.33
Stem+Suffix/Meta	59.25	61.17	57.24	60.30	54.77	58.87	58.60
Lemma+Suffix/Meta	58.77	60.97	56.59	58.15	53.75	58.40	57.77
Token-Meta	55.57	57.48	55.94	46.35	53.75	54.55	53.94

Table 4.15: LSTM Accuracy Results w/ Majority Voting

	Movie	Book	DVD	Electr.	Kitchen	Beyaz Perde	Avg
Word-Token	**90.80**	77.86	76.43	**80.71**	75.00	87.39	**81.36**
BPE-5k	89.86	75.71	75.71	79.29	75.00	**87.91**	80.58
Lemma	89.20	80.00	77.86	75.00	75.00	86.20	80.54
BPE-30k	89.30	72.14	**79.29**	74.29	**77.14**	88.56	80.12
Stem	89.39	78.57	78.57	78.57	68.57	86.28	79.99
Hybrid	89.48	75.00	73.57	80.00	74.29	86.67	79.83
Stem+Suffix	90.05	80.71	74.29	76.43	70.71	86.80	79.83
Lemma+Suffix	90.33	77.86	**79.29**	77.14	65.71	87.06	79.56
Syllable	87.23	**82.14**	70.71	75.00	75.00	85.61	79.28
BPE-1k	86.48	70.00	72.14	73.57	67.14	85.43	75.79
Character	72.58	60.71	55.00	64.29	60.00	81.37	65.66
Stem+Suffix/Meta	65.26	65.71	64.29	65.00	57.86	67.43	64.26
Lemma+Suffix/Meta	64.23	68.57	58.57	62.86	61.43	66.00	63.61
Token-Meta	57.37	58.57	64.29	45.71	52.86	57.94	56.12

tent throughout the comparison, we will make segmentation method selection based on overall average scores and not individual scores for datasets we are making the comparison.

Following segmentation methods achieve top 3 highest average scores among classifiers.

- ***CNN***: Based on Table 4.13
  - ***Lemma+Suffix***: Avg 81.27%
  - ***BPE-5k***: Avg 80.70%
  - ***Hybrid***: Avg 80.59%
- ***LSTM***: Based on Table 4.15
  - ***Word-Token***: Avg 81.36%
  - ***BPE-5k***: Avg 80.58%
  - ***Lemma***: Avg 80.54%

We will present our result sets in the form of $<$ *model* $>$ @ $<$ *segmentation* $>$ for simplicity. Each of these sets will be called a classifier, since a segmentation alongside a neural network model constitutes a classifier for Turkish text input.

In Table 4.16, we present our results on *Movie Reviews* in comparison to two previous studies in the literature, SentiStrength in Vural et al. (2012) and Feature Selection in Firat Akba and Sever (2014). The table shows that both our selected CNN and LSTM results outperform the best results in baseline scores, which belong to Firat Akba and Sever (2014). On average our scores are +3 points higher than results in Firat Akba and Sever (2014). Our best score, LSTM @ *BPE-5k*, outperforms it by a large +4 point margin.

We also present our results on *Product Reviews* in Table 4.17 in comparison to earlier scores derived from the dataset by Demirtas and Pechenizkiy (2013). Our results outperform baseline scores in all categories. In average, the margin between our results and best baseline scores is 6.7.

Table 4.16: Accuracy Scores on the *Movie Reviews* data set

Model	Author	**Acc.**
Sentence-binary	Vural et al. (2012)	70.39
Sentence-max/min	Vural et al. (2012)	74.83
Word-sum	Vural et al. (2012)	75.90
Chi Square	Firat Akba and Sever (2014)	**83.90**
Information Gain	Firat Akba and Sever (2014)	**83.90**
CNN @ *Lemma+Suffix*		86.30
CNN @ *BPE-5k*		86.46
CNN @ *Hybrid*		85.96
LSTM @ *Word-Token*		87.39
LSTM @ *BPE-5k*		**87.91**
LSTM @ *Lemma*		86.20

In fact, we have a single classifier that outperforms all baseline scores alone, which is CNN @ *BPE-5k*. This classifier actually holds another impressive achievement. It gets to be at top 3 selected result set in both CNN and LSTM models, holding the second position in both. In Table 4.18 we present overall margins between our results and best baseline scores. The table shows that CNN @ *BPE-5k* classifier outperforms baseline scores on an average of +5.5 points. Consolidated results from selected classifiers outperform baseline scores on an average of +7.92 points and Overall results on an average of +8.21 points.

The motivation behind our work originally did not get higher results than baseline scores. It rather was to compare and measure performances of different segmentation methods and neural network models. However, accomplishing something like this is also worth mentioning.

4.6 Performance Evaluation

In this part, we will evaluate performances of segmentation methods, and neural networks and investigate into what they do good and what they could do better.

Table 4.17: Accuracy Scores on the *Product Reviews* data set

Model	**Movie**	**Book**	**DVD**	**Electronics**	**Kitchen**
Naive Bayes[1]	69.50	72.40	76.00	73.00	75.90
Naive Bayes MT[1]	70.00	71.70	74.90	64.40	69.60
Linear SVC[1]	66.00	66.60	70.30	72.40	70.00
Linear SVC MT[1]	66.50	66.90	67.60	64.40	67.30
CNN @ *Lemma+Suffix*	90.61	**81.43**	76.43	77.86	75.00
CNN @ *BPE-5k*	90.61	75.71	76.43	75.00	**80.00**
CNN @ *Hybrid*	89.01	75.71	75.00	**80.71**	77.14
LSTM @ *Word-Token*	**90.80**	77.86	76.43	**80.71**	75.00
LSTM @ *BPE-5k*	89.86	75.71	75.71	79.29	75.00
LSTM @ *Lemma*	89.20	80.00	**77.86**	75.00	75.00

[1]Demirtas and Pechenizkiy (2013)

Table 4.18: Margin between best baseline scores and our results on *Movie Reviews*, and *Product Reviews* data sets

	Movie	Book	DVD	Electr.	Kitchen	Beyaz Perde	**Avg**
Overall	**+21.1**	**+9.8**	**+3.3**	**+7.7**	**+4.1**	**+4.0**	**+8.3**
Selected Models	+20.8	+9.0	+1.9	**+7.7**	**+4.1**	**+4.0**	+7.9
CNN @ *BPE-5k*	+20.6	+3.3	+0.4	+2.0	**+4.1**	+2.6	+5.5

4.6.1 Distribution of Polarity Predictions and Majority Voting

Text classification is a binary problem. Therefore, neural networks are trained to output a value between 0-1 according to how certain the model is about the polarity of the input text. As a rule of thumb, values near 0.5 are those uncertain and they are very likely to be neutral. Similarly, a prediction near 1 or 0 is one the model is pretty sure that it is negative or positive. For instance, a prediction value 0.96 will indicate that model thinks the input text is almost certainly positive.

In order to evaluate performances of models, we will first compile entire predictions into one single prediction file. From there, we can extract distribution of a particular neural network with a particular segmentation method. Since only *BPE-5k* was able to take its place among selected segmentation methods both for CNN and LSTM, and also because CNN @ *BPE-5k* literally outperforms any baseline score, we will use *BPE-5k* as our benchmark segmentation model. However, we will occasionally switch from CNN to LSTM when needed.

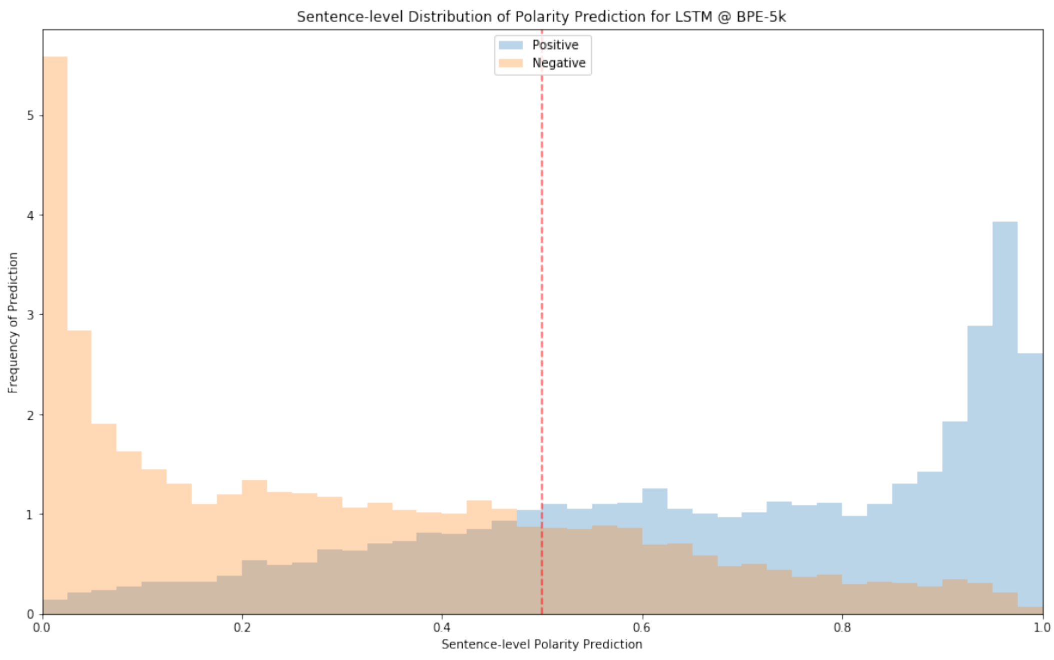
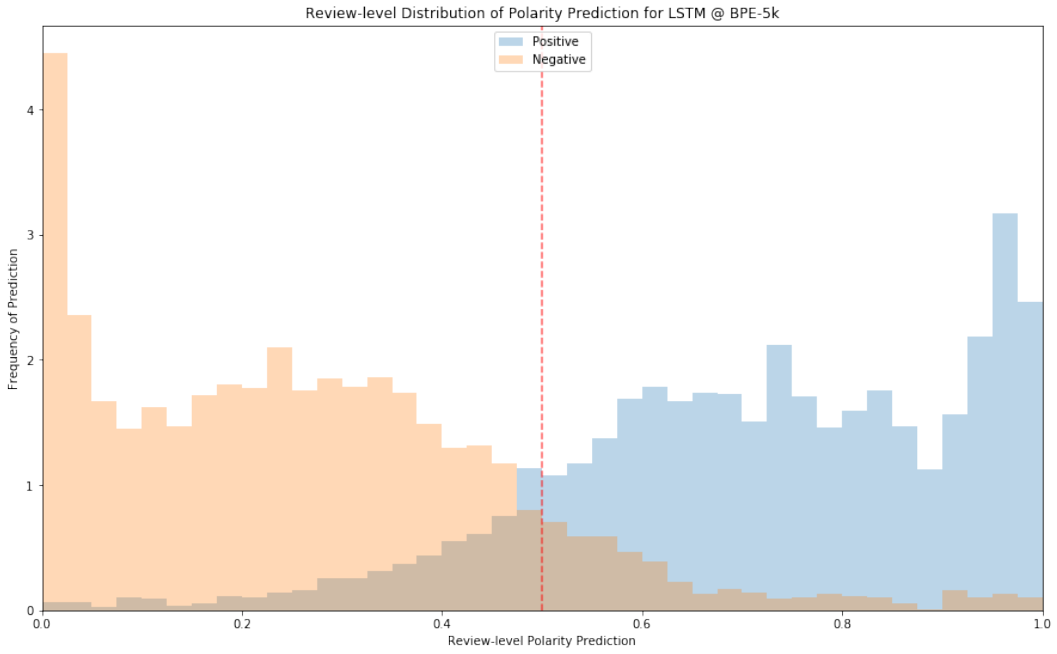


Figure 4.5: Predicted polarity distribution of test sample data for an LSTM networks trained with *BPE-5k* segmentation output. Distribution is Sentence-level.

Figure 4.5 shows the distribution of polarity prediction by our LSTM network trained by *BPE-5k* segmentation output. Orange area shows the distribution of prediction for sentences labeled as Negative, and blue area shows the distribution of sentences labeled as Positive. One important aspect in this figure is that it is similar to a Chi-squared distribution with $k = 2$. However, there are too many values in between, possibly due to data not being clean enough. Hence, this is overall an expected distribution. Our model is specifically trained to output binary values for classification.

The overlapping areas constitute samples classified incorrectly. From average accuracy score of *BPE-5k*, we can deduct the ratio of the overlapping area. Table 4.14 indicates that the average score for *BPE-5k* is 71.56, which means that the overlapping area constitutes $100 - 71.56 = 28.44\%$ of overall distribution.

Figure 4.6: Predicted polarity distribution of test sample data for an LSTM networks trained with *BPE-5k* segmentation output. Distribution is Review-level



In Figure 4.6 we can see the distribution of review-level polarity prediction frequency distribution. In this case, the distribution is normal in comparison to the initial distribution. This is also an expected behavior if we take into account how the majority voting process works.

During pre-processing we divided reviews into sentences only to be compiled into the same review after polarity predictions are derived from neural network classifier. The compilation works in a way that a small sample of sentences is taken from sentence sample and the combined average predicted value is calculated for the review. The procedure we are describing here is the one that *Central Limit Theorem* (CLT) adheres to explain. CLT indicates that, when a sample size of n is drawn repeatedly from a distribution, the distribution of means of these samples will have a standard deviation (σ) value of square root times the σ of the original distribution. On the other hand, It will still have the same mean value ofμ. See Equation 4.1 for reference.

$$\sigma_n = \frac{\sigma_0}{\sqrt{n}} \tag{4.1}$$

This means that the new distribution will be converging to a normal distribution with smaller standard deviation around the same mean. Majority voting creates a new distribution in the same way, and therefore, expected to have a normal distribution around mean value of the original distribution.

Figure 4.7 shows the distribution of polarity prediction by our CNN network trained

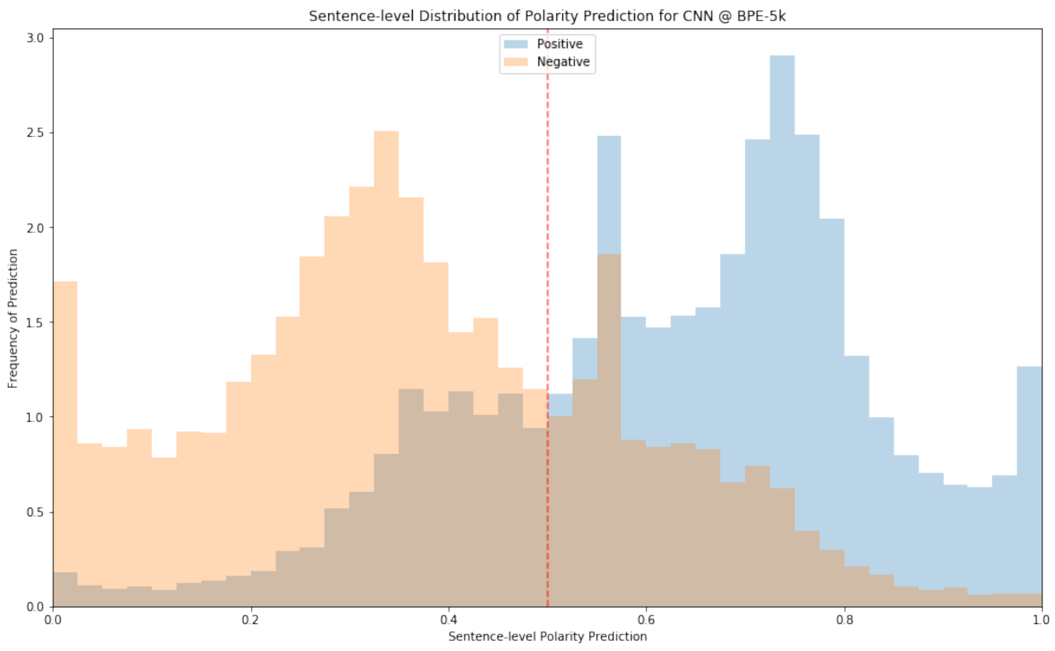


Figure 4.7: Predicted polarity distribution of test sample data for a CNN networks trained with *BPE-5k* segmentation output. Distribution is Sentence-level

by *BPE-5k* segmentation output. This distribution is quite different than the LSTM sentence-level distribution. What is quite surprising about it is the fact that distribution is a normal distribution with an excess accumulation at the outer edges. This is strange because in a way it means the CNN network is not very certain about the polarity of the majority of test entries fed into the model. The overlapping area is also accumulated around the neutral region. This area is more spread-out in LSTM distribution.

In Figure 4.8 we can see the distribution of review-level polarity prediction frequency distribution for CNN network. Due to the same reasons that we listed for LSTM review-level distribution, the new sample space will have a more accumulated normal distribution thanks to Central Limit Theorem.

4.6.2 Performance Comparison on Sample Cases

Table 4.19 shows a sample prediction for a random review extracted from predictions derived from LSTM @ *BPE-5k* and LSTM @ *Lemma+Suffix*. The review constitutes 4 different sentences. The LSTM @ *BPE-5k* classifier provided 0.36, 0.23, 0.76, and 0.78 scores respectively for each sentence. The mean value is 0.53, and larger than 0.5. Therefore, we can conclude that the review is classified correctly by a small margin. The LSTM @ *Lemma+Suffix* classifier, however, provides much better scores than the former. It derives 0.83, 0.79, 0.88, and 0.21 scores for sentences respectively. The average is 0.67, and unlike *BPE-5k* the review is classified as positive by a larger margin.

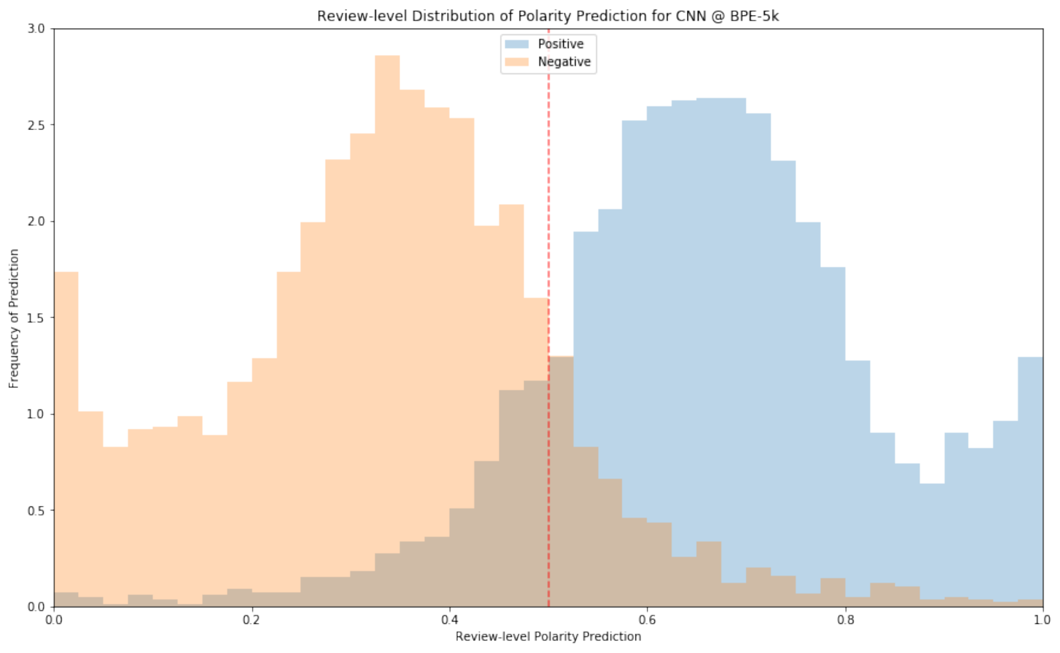


Figure 4.8: Predicted polarity distribution of test sample data for a CNN networks trained with *BPE-5k* segmentation output. Distribution is Review-level

Table 4.19: Sample Predictions for a review compiled from LSTM @ *BPE-5k* and LSTM @ *Lemma+Suffix*

Sentence	Label	C1[1]	C2[2]
elif şafakın imzalı kitabını hepsiburadan temin etmek çok güzel . (being able to acquire elif şafak's[typo] signed book from hepsiburada[typo] is very nice .)	1.0	0.36	0.83
bu fırsat kaçırılmamalı . (this opportunity shouldn't be missed .)	1.0	0.23	0.79
hediye olarak çok güzel bir kitap . (this is a very good book for a present .)	1.0	0.77	0.88
konudan bahsetmeyeceğim , elif şafakı tanıyanlar bilir ... (i won't talk about the theme , those that know elif şafak(typo) will know ...)	1.0	0.78	0.21

[1]LSTM @ *BPE-5k*
[2]LSTM @ *Lemma+Suffix*

From a manual inspection, we can conclude that the review is clearly positive, and the author or review is praising the author of the book s/he is reviewing. The distinction is important in a way it can distinguish what a morphological analysis such as *Lemma+Suffix* can achieve better than a sub-word model such as *BPE-5k*. However, we also know that *BPE-5k* has a better overall performance than *Lemma+Suffix*. Therefore we can conclude that there are many factors at play, and the overall schema opts for the *BPE-5k*.

Table 4.20: Sample Predictions for a review compiled from CNN @ *BPE-5k* and CNN @ *Lemma+Suffix*

Sentence	Label	C1[1]	C2[2]
quantin tarantino iyi bir yönetmen ama ben bu kadar kötü bir film beklemezdim . (quantin tarantino is a good director, but i wouldn't expect a movie this bad .)	0.0	0.23	0.24
film çok sıkıcı başladı inatla sonuna kadar izledim acaba değişir mi diye ama hüsran . (movie started dull, i resisted to the end hoping it would change, but disappointment .)	0.0	0.11	0.04
bana göre gerçekten kötü bir film (to me it is a really bad movie)	0.0	0.29	0.15

[1]CNN @ *BPE-5k*
[2]CNN @ *Lemma+Suffix*

Table 4.20 shows sentiment predictions by CNN with *BPE-5k* and *Lemma+Suffix* segmentations models for a sample review. Table shows that the negative review has an average sentiment score of 0.21 and 0.14 for *BPE-5k* and *Lemma+Suffix* respectively. We can once again conclude that even if *BPE-5k* can be more robust than most other candidates in terms of accuracy, the sentiment prediction is more certain with *Lemma+Suffix*, and thereby with morphological approaches.

4.6.3 Cross-parameter Comparison

In order to understand the relationship between different parameters and their effect on accuracy results, we compiled various parameters for performance evaluation during model training.

During our experiments, we collected a wide range of measurements on preprocessing, training, and evaluation. These results are available both for CNN and LSTM experiments. The parameters measured and collected are shown in Table 4.21 with their descriptions.

In Figure 4.9 we can see the relationship between majority voting accuracy, and vocabulary. The details Segmentation and Review length parameters add further insight. We can see that there is a clear correlation between vocabulary size and accuracy. However, the relation is possibly non-linear, and the accuracy will start to deteriorate after a vocabulary size. In fact, our most promising segmentation *BPE-5k* has a vocabulary size of 5k. This figure itself also hints at deteriorating nature of this

Table 4.21: List of parameters collected during experiments with pre-precessing and neural model building.

Parameter	Description
No	The order in which the experiment is executed.
Dataset	Name of dataset
Segmentation	Segmentation method used during preprocessing
Train	Size of training sample
Validation	Size of validation sample
Test	Size of test sample
Batch Size	Batch size used model training*
Vocabulary	Vocabulary size
Max Review Length	Size of longest review in dataset after pre-processing
Pre-processing Duration	Pre-processing duration in seconds
Train Duration	Model training duration in seconds
Evaluation Duration	Test data prediction duration
Score	Sentence-level raw score
MV Score	Review-level score after Majority Voting
Epoch Count	Number of epochs run until early stopping kicks in
Save Epoch	The epoch number the best model is encountered and saved

* Batch size is calculated after preprocessing. The final batch size depends on vocabulary and maximum review length.

correlation around 20k vocabulary size. However, further research will be needed to prove this claim. We will not go into further detail.

4.6.4 Computational Performance

In this part, we will address three fundamental performance metrics. These metrics are of a high value in order to express a definitive way of dealing with sentiment analysis task in Turkish texts on a service level. An efficient way of dealing with this task is creating the most value with the least investment. Therefore, we will be investigating the performance of each classifier alongside the resources these uses during sentiment extraction.

In order to determine the resources model use, we identified a set of metrics, and

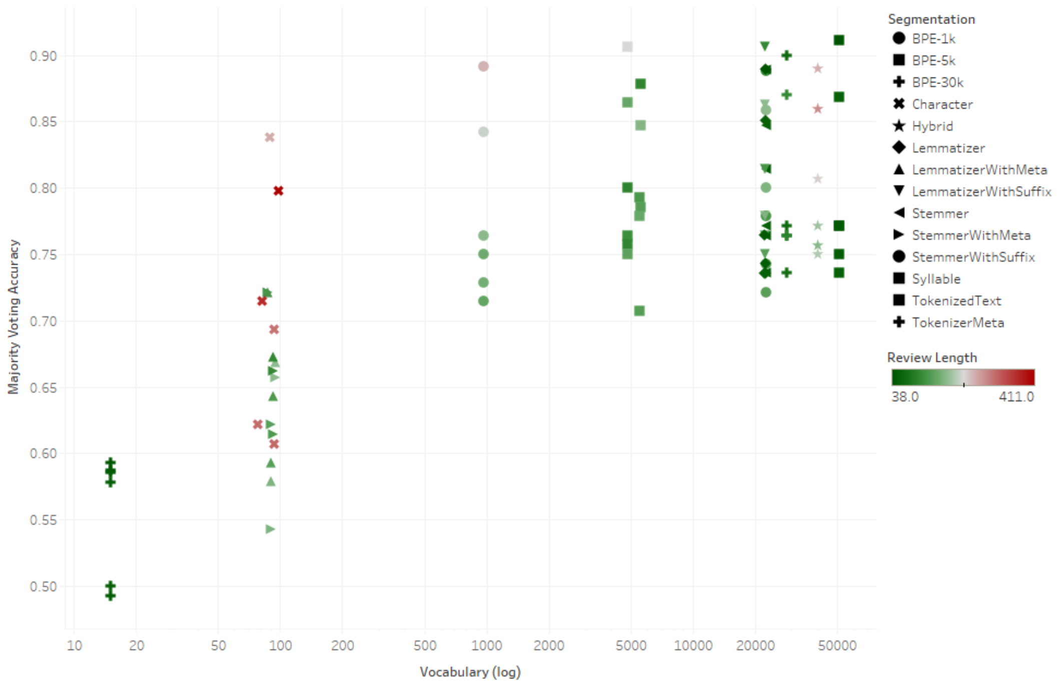


Figure 4.9: Scatter plot showing relation between vocabulary size and accuracy among all results acquired with CNN. Review length and Segmentation are added as details.

kept measurements regarding these metrics. In order to measure memory allocation, we calculated the total memory neural network needs in order to run. Since neural networks built by Keras running on Tensorflow back-end are able to use Graphics Processing Unit (GPU) the memory allocation here will represent GPU memory used for the neural network. On the other hand, we also determined the parameters affecting the total memory neural networks allocate to run. Formulas 4.2 and 4.3 show how to calculate total memory a CNN or an LSTM network allocate based on parameters such as vocabulary and maximum input size.

On the other hand, in order to calculate processing power allocated to the classifiers we identified the duration of calculations as the candidate. Our implementation uses there processing steps in conjunction with each other in order to pre-process data, train a model based on this, and test the model with test subset. The code designed to do this also keeps track of durations all these steps take. The first time-stamp (t_0) is recorded during the initialization, where the dataset is to be pushed into data pre-processing unit. The second time-stamp (t_1) is recorded when the pre-processing is complete. The difference between the two ($t_1 - t_0$) provides pre-processing duration. After this point, model building and training starts. The third time-stamp (t_2) is recorded after training stage is complete. The difference between third and second time-stamps ($t_2 - t_1$) gives the model training duration. Finally, test data subset is fed into the trained model for sentiment predictions. When prediction stage is complete, the last time-stamp (t_3) is recorded. The difference between the last and third time-stamps ($t_3 - t_2$) gives the evaluation duration.

The first and the most important metric is the accuracy, which we covered throughout the report. Without a plausible accuracy rate, a faster classifier with a smaller footprint will be just as meaningless. We must first explore and discover an efficient way of achieving a good accuracy in order to go on with computational performance related issues.

The second one is how much resources it takes up during training and evaluation. This is also crucial if we are going to deploy the implementation and neural modal for regular usage. The resources could be considered in two main groups, memory and computation.

The third one is how fast it is with extracting sentiment from a sample input text.

The First performance metric is heavily addressed in earlier parts, therefore, we will only add it to discussions as part of investigation detail. Resource consumptions and Prediction speed will be primary topics. We will use data described in Table 4.21 for performance evaluations in this part.

Not all parameters we need exists within our data, some of them will be calculated based on available parameters. We will create parameters for memory consumed by CNN and LSTM networks, and training duration until the best model is encountered.

The new parameters can be defined as follows.

$$M_{CNN} = 50 * v + 500 * l + 3121 \tag{4.2}$$

$$M_{LSTM} = 32 * v + 0 * l + 53301 \tag{4.3}$$

$$t_{atd} = t_{training} * (1 - (EC - SE)/EC) \tag{4.4}$$

$$t_{te} = t_{pp} + t_{eval} \tag{4.5}$$

Where;

v denotes *Vocabulary size for preprocessed dataset*,

l denotes *Max Review Length for neural network model input*,

M_{CNN} denotes *Amount of Memory the CNN model uses*,

M_{LSTM} denotes *Amount of memory the LSTM model uses*,

$t_{training}$ denotes *The duration needed to train the network*,

EC denotes *Total amount of Epochs until the training is finalized*,

SE denotes *The epoch on which the best results encountered during training*,

t_{atd} denotes *Actual Train Duration: Calculated training duration until best results are encountered*,

t_{pp} denotes *Pre-processing Duration: The time it takes to process raw input into the shape the network can process*,

t_{eval} denotes *Evaluation Duration: The time it takes to evaluate a preprocessed input*,

t_{te} denotes *Total Evaluation Duration: Total amount of time it take to pre-process raw data and evaluate it.*

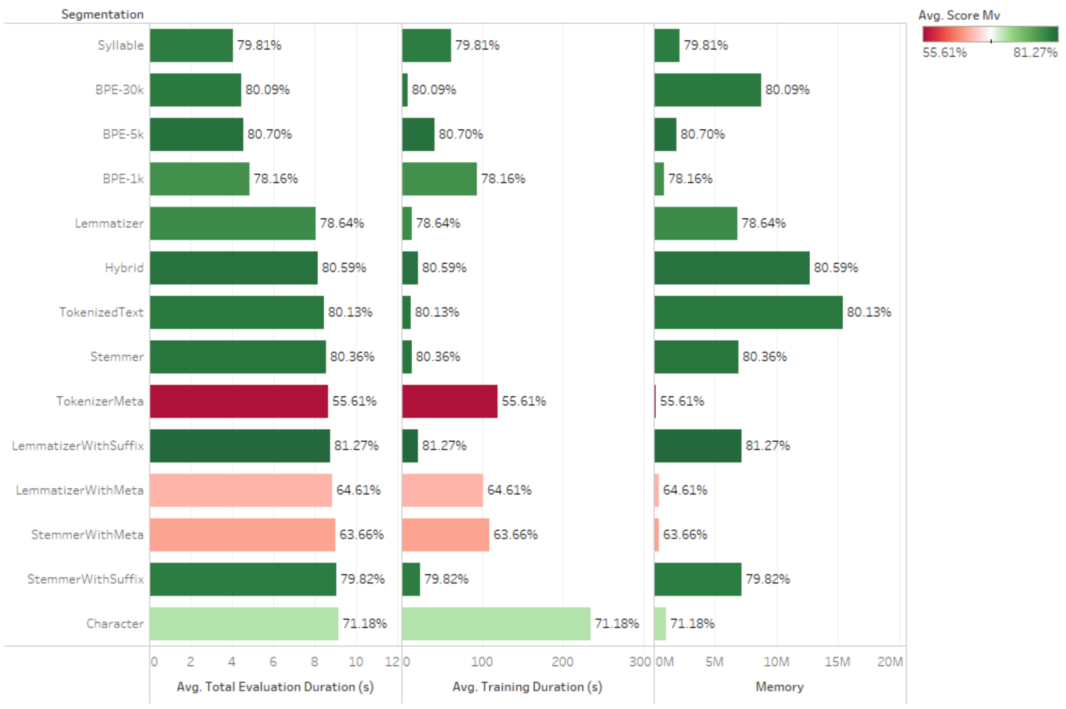


Figure 4.10: Bar chart showing average memory size, average test data prediction duration, and model memory for a CNN network.

We extracted the new parameters by using formulas provided in 4.2, 4.3, 4.4, and 4.5. Using new parameters we plotted the relationships in column charts both for Evaluation Duration and Training Duration. Figure 4.10 shows the distribution of these two parameters among different segmentation methods. The columns are sorted by ascending average evaluation duration.

From the figure we can deduct that *Syllable*, *BPE-1k*, *BPE-5k*, *BPE-30k* methods provide shortest total evaluation durations. Note that *Total Evaluation Duration* is the sum of test data pre-processing, and prediction durations. On the other hand, they also hold some of highest scores for accuracies. *BPE-5k* seems to be an obvious candidate for the most efficient model in terms of accuracy, training duration, and memory.

4.7 Summary

The results we acquired during experiments show that there is substantial support for nominating *BPE-5k* segmentation method as the most efficient one combining our metrics both for accuracy and computational performance. However, one important thing to point out is the fact that the accuracy results among prominent segmentation methods are not significant. There is also strong evidence that the performances of CNN and LSTM networks seem to be similar.

The following points can be deducted from results presented in this work.

- *BPE-5k* Seems to be performing good both in terms of accuracy and computational performance.
- CNN and LSTM networks perform similarly.
- Widely used *Word-Token* segmentation method performs as good as other ones with similar scores in terms of accuracy.
- Unlike accuracy, performance evaluation results for computational metrics vary to a great extent for different segmentation methods. There is strong evidence that this difference should be taken into consideration in model selection.
- The positional attributes of words in sentences without word themselves keeps a substantial amount of info by which these models achieved accuracy results around 65%. This also means that this information could be used as an additional layer in future.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Our results clearly show that neural network models are more successful as carefully engineered lexicon and rule-based methods, mostly by a large margin. We selected the best performing classifiers among CNN and LSTM networks. *Lemma+Suffix*, *BPE-5k*, *Hybrid*, *Word-Token*, and *Lemma* proven to be the best ones. Our selected models outperformed best scores in baselines by an average +7.92 points. (Tables 4.17, 4.18, and 4.16)

We also noted *BPE-5k* model for its performance with being among top 3 for results acquired by using both CNN and LSTM based classifiers. It also achieves better scores than any baseline score alone. This model outperformed best scores in baselines by an average +5.5 points along all datasets. (Table 4.18)

Our prediction distributions showed that LSTM networks seem to be more certain about their predictions with a Chi-squared distribution. (Figure 4.5) Whereas, CNN distribution is normal near neutral region. (Figure 4.7)

We also observed Central Limit Theorem being at play with our Majority Voting process. We also validated the effect with CLT formulas. (Equation 4.1)

We reviewed some sample outputs and their predictions by different segmentation methods. We observed that even if they predict the same output, they do it for seemingly quite different reasons, which showed that there are very complex factors being at play for different segmentation methods. (Table 4.19)

We also investigated the relationships between model scores, segmentation methods and review lengths. We observed that the networks performed better until a certain vocabulary size. (Figure 4.9)

We evaluated computational performances of models and segmentation methods. We derived new parameters and created various plots to observe differences between performances in terms of memory usage, average training duration, and average input evaluation duration. We concluded once again that *BPE-5k* performs very well for all metrics. We indicated this model as the final candidate for text classification in Turkish. (Figure 4.10)

This work did not focus on advanced natural language processing techniques such as phrase and idiom extraction, and sentence attention detection. On the other hand, it does not try to extract sentence forms and types. We believe that, by employing such methods, scores acquired in this work could be improved. In addition, by inspecting the predictions which provide poor accuracy the more important features could be determined. Thus, it could be possible prioritize these advanced methods.

In this work, we also did not try to check and correct typographical errors within the text. Informal texts are known to be hosting too many errors of this type. By employing spell-checking and advanced error correction mechanisms vocabulary could be reduced and accuracies could be improved. In addition, text can be normalized where information is formulated or represented in other forms such as numbers. Normalization might both reduce vocabulary size and increase accuracy. We think that using typo checkers and text normalization can add further improvement to our models and accuracy scores.

We also did not intend to use advanced disambiguation for words while extraction positional attributes and suffixes. Turkish has too many seemingly similar or the same but structurally different words due to its rich morphology. Developing and utilizing a tool capable of understanding the context of the sentence and determining the most appropriate variant of these words could improve accuracy performances our models further. For instance, the word *kara* (dark; land; into snow) is used with different meanings in following phrases.

- **Kara** bulutlar gökleri doldurdu. (**Dark** clouds filled the skies.)
- Ada Marmarada'ki ufak bir **kara** parçasından oluşmaktadır. (The island is a small **land** in Marmara.)
- **Kara** batan ayaklarını hızlıca çekti. (He quickly pulled his legs which were sinking **into snow**.)

As already mentioned in Sentiment Analysis with Various Segmentation Methods and Deep Learning Models chapter (Part 3.2.5) LSTM networks are capable of processing multiple layers of data. This means data processed in different ways can be used to create data in different forms each of which represents different attributes of the data. These different sets of attributes could be used as different layers. This means various segmentation methods presented in this work could be assigned as different layers to a new multi-layered LSTM network. We believe such an arrangement could improve the performance of the sentiment analysis task.

Finally, we did not use WordVec libraries in order to build embeddings for neural network models we used due to several reasons pointed out in introduction chapter. The most important of these reasons was the fact that our segmentation methods divide each word into sub-word fragments which by themselves mostly do not have a meaning or a meaning related to the word used. However, it is still possible to build a WordVec for each segmentation method by calculating weights for each of these fragments by building a model that will calculate the ratio of each fragment in various words. The eventual weights could be calculated with basic arithmetic formulas.

Bibliography

Mehmet Dundar Akin Ahmet Afsin Akin. Zemberek, an open source nlp framework for turkic languages. 2007. doi: 10.1.1.556.69.

Rıfat Aşlıyan and Korhan Günel. Design and implementation for extracting turkish syllables and analyzing turkish syllables. 10 2007.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *in Proc. of LREC*, 2010.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

Zeynep Boynukalin. Emotion analysis of turkish texts by using machine learning methods. Master's thesis, Middle East Technical University, 2012.

Erik Cambria, Daniel Olsher, and Dheeraj Rajagopal. Senticnet 3: A common and common-sense knowledge base for cognition-driven sentiment analysis. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1515–1521. AAAI Press, 2014. URL `http://dl.acm.org/citation.cfm?id=2892753.2892763`.

Onder Coban, Baris Ozyer, and Gulsah Tumuklu Ozyer. Sentiment analysis for turkish twitter feeds. In *2015 23nd Signal Processing and Communications Applications Conference (SIU)*. Institute of Electrical and Electronics Engineers (IEEE), may 2015. doi: 10.1109/siu.2015.7130362.

Ça Çoltekin, Cem Bozşahin, et al. Syllables, morphemes and bayesian computational models of acquiring a word grammar. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 29, 2007.

Çağrı Çöltekin. A set of open source tools for turkish natural language processing. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA), 2014. URL `http://www.aclweb.org/anthology/L14-1375`.

Rahim Dehkharghani, Yucel Saygin, Berrin Yanikoglu, and Kemal Oflazer. Sentiturknet: a turkish polarity lexicon for sentiment analysis. *Language Resources and Evaluation*, 50(3):667–685, 2016a.

Rahim Dehkharghani, BERRIN YANIKOGLU, YUCEL SAYGIN, and Kemal Oflazer. Sentiment analysis in turkish at different granularity levels. *Natural Language Engineering*, pages 1–25, 2016b.

Hakan Demir and Arzucan Özgür. Improving named entity recognition for morphologically rich languages using word embeddings. In *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*, pages 117–122. IEEE, 2014.

Erkin Demirtas and Mykola Pechenizkiy. Cross-lingual polarity detection with machine translation. In *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining - WISDOM13*. Association for Computing Machinery, 2013. doi: 10.1145/2502069.2502078. URL `http://sentic.net/wisdom2013pechenizkiy.pdf`.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. ISSN 1551-6709. doi: 10.1207/s15516709cog1402_1. URL `http://dx.doi.org/10.1207/s15516709cog1402_1`.

Umut Erogul. Sentiment analysis in turkish. Master's thesis, Middle East Technical University, 2009.

Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC-06*, pages 417–422, 2006.

Ebru Akcapinar Sezer Firat Akba, Alaettin Ucan and Hayri Sever. Assesment of feature selection metrics for sentiment analysis: Turkish movie reviews. 2014.

Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *NEURAL COMPUTATION*, 12:2451–2471, 1999.

Anastasia Giachanou and Fabio Crestani. Like it or not: A survey of twitter sentiment analysis methods. *ACM Comput. Surv.*, 49(2):28:1–28:41, June 2016. ISSN 0360-0300. doi: 10.1145/2938640. URL `http://doi.acm.org/10.1145/2938640`.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013. URL `http://arxiv.org/abs/1303.5778`.

Michiel Hermans and Benjamin Schrauwen. Training and analysing deep recurrent neural networks. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 190–198. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/5166-training-and-analysing-deep-recurrent-neural-networks.pdf`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

Ozan Irsoy and Claire Cardie. Deep recursive neural networks for compositionality in language. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2096–2104, 2014a. URL `http://papers.nips.cc/paper/5551-deep-recursive-neural-networks-for-compositionality-in-language`.

Ozan Irsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 720–728. ACL, 2014b. ISBN 978-1-937284-96-1. URL `http://aclweb.org/anthology/D/D14/D14-1080.pdf`.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June 2014.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics (ACL), 2014. doi: 10.3115/v1/d14-1181.

Kezban Dilek Kisa and Pinar Karagoz. Named entity recognition from scratch on social media. In *Proceedings of the 6th International Workshop on Mining Ubiquitous and Social Environments (MUSE 2015) co-located with the 26th European Conference on Machine Learning / 19th European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2015), Porto, Portugal, September 7, 2015.*, pages 2–17, 2015. URL `http://ceur-ws.org/Vol-1521/paper2.pdf`.

Nadin Kokciyan, Arda Celebi, Arzucan Ozgur, and Suzan Uskudarli. Bounce: Sentiment classification in twitter using rich feature sets. Citeseer, 2013.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. Charner: Character-level named entity recognition. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 911–921, 2016.

Hugo Larochelle and Geoffrey E Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1243–1251. Curran Associates, Inc., 2010. URL `http://papers.nips.cc/paper/4089-learning-to-combine-foveal-glimpses-with-a-third-order-boltzmann-machine.pdf`.

Rémi Lebret and Ronan Collobert. N-gram-based low-dimensional representation for document classification. *CoRR*, abs/1412.6277, 2014. URL `http://arxiv.org/abs/1412.6277`.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *CoRR*, abs/1610.03017, 2016. URL `http://arxiv.org/abs/1610.03017`.

Jiwei Li, Dan Jurafsky, and Eduard H. Hovy. When are tree structures necessary for deep learning of representations? *CoRR*, abs/1503.00185, 2015. URL `http://arxiv.org/abs/1503.00185`.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. URL `http://arxiv.org/abs/1508.04025`.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9. URL `http://dl.acm.org/citation.cfm?id=2002472.2002491`.

George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11): 39–41, November 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL `http://doi.acm.org/10.1145/219717.219748`.

B. Pang and L. Lee. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. *eprint arXiv:cs/0409058*, September 2004.

Colin Raffel and Daniel P. W. Ellis. Feed-forward networks with attention can solve some long-term memory problems. *CoRR*, abs/1512.08756, 2015. URL `http://arxiv.org/abs/1512.08756`.

M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997. ISSN 1053-587X. doi: 10.1109/78.650093. URL `http://dx.doi.org/10.1109/78.650093`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015. URL `http://arxiv.org/abs/1508.07909`.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2011.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics, 2012.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D13-1170`.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558, 2010. ISSN 1532-2890. doi: 10.1002/asi.21416. URL `http://dx.doi.org/10.1002/asi.21416`.

Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *CoRR*, abs/1003.1141, 2010. URL `http://arxiv.org/abs/1003.1141`.

A. Gural Vural, B. Barla Cambazoglu, Pinar Senkul, and Z. Ozge Tokgoz. A framework for sentiment analysis in turkish: Application to polarity detection of movie reviews in turkish. In *Computer and Information Sciences III*, pages 437–445. Springer Nature, oct 2012. doi: 10.1007/978-1-4471-4594-3_45.

Mengqiu Wang, Rob Voigt, and Christopher D. Manning. Two knives cut better than one: Chinese word segmentation with dual decomposition. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics (ACL), 2014. doi: 10.3115/v1/p14-2032.

Ezgi Yildirimm, Fatih Samet Cetin, Gulsen Eryigit, and Tanel Temel. The impact of nlp on turkish sentiment analysis. *Turkiye Bilisim Vakfi Bilgisayar Bilimleri ve Muhendisligi Dergisi*, 7(1 (Basili 8), 2015.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of CNN and RNN for natural language processing. *CoRR*, abs/1702.01923, 2017. URL `http://arxiv.org/abs/1702.01923`.

Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015. URL `http://arxiv.org/abs/1510.03820`.

TEZ FOTOKOPİ İZİN FORMU

<u>ENSTİTÜ</u>

Fen Bilimleri Enstitüsü	☐
Sosyal Bilimler Enstitüsü	☐
Uygulamalı Matematik Enstitüsü	☐
Enformatik Enstitüsü	☐
Deniz Bilimleri Enstitüsü	☐

<u>YAZARIN</u>

Soyadı : ....
Adı : ....
Bölümü : ....

<u>TEZİN ADI</u> (İngilizce) : ....
....
....
....
....

<u>TEZİN TÜRÜ</u> : Yüksek Lisans ☐ Doktora ☐

1. Tezimin tamamı dünya çapında erişime açılsın ve kaynak gösterilmek şartıyla tezimin bir kısmı veya tamamının fotokopisi alınsın. ☐

2. Tezimin tamamı yalnızca Orta Doğu Teknik Üniversitesi kullancılarının erişimine açılsın. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.) ☐

3. Tezim bir (1) yıl süreyle erişime kapalı olsun. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.) ☐

Yazarın imzası .... Tarih ....