REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

RECURSIVE DEEP LEARNING FOR TURKISH SENTIMENT ANALYSIS

Sultan ZEYBEK

DOCTOR OF PHILOSOPHY THESIS
Department of Mathematical Engineering
Program of Mathematical Engineering

Supervisor
Prof. Dr. Aydın SEÇER

Co-supervisor
Assist. Prof. Dr. Ebubekir KOÇ

August, 2021

REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

RECURSIVE DEEP LEARNING FOR TURKISH SENTIMENT ANALYSIS

A thesis submitted by Sultan ZEYBEK in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY** is approved by the committee on 23.08.2021 in Department of Mathematical Engineering, Program of Mathematical Engineering.

Prof. Dr. Aydın SEÇER
Yildiz Technical University
Supervisor

Assist. Prof. Dr. Ebubekir KOÇ
Fatih Sultan Mehmet Vakif University
Co-supervisor

Approved By the Examining Committee

Prof. Dr. Aydın SEÇER, Supervisor
Yildiz Technical University _____

Prof. Dr. İbrahim EMİROĞLU, Member
Yildiz Technical University _____

Prof. Dr. Mustafa BAYRAM, Member
Biruni University _____

Prof. Dr. Samet Yücel KADIOĞLU, Member
Istanbul Technical University _____

Assoc. Prof. Dr. Birol ASLANYÜREK, Member
Yildiz Technical University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of Recursive Deep Learning for Turkish Sentiment Analysis supervised by my supervisor, Prof. Dr. Aydın SEÇER. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Sultan ZEYBEK

Signature

This study was supported by The Scientific and Technological Research Council of Turkey (TUBITAK), 2214-A International Research Fellowship Programme (for PhD Students), Grant No. 1059B141800193.

Dedicated to my family
for their endless love and unconditional support

ACKNOWLEDGEMENTS

I would like to thank you my supervisor Prof. Dr. Aydın Seçer, who has always been supportive to me throughout my PhD journey. I would like to thank my co-supervisor Dr. Ebubekir Koç who always supports me and opens my way to the new opportunities. Thanks to him, I met Professor Duc Truong Pham and became a permanent member of the beehive. I would like to thank the members of my thesis committee Prof. Dr. Mustafa Bayram and Prof. Dr. İbrahim Emiroğlu, and also many thanks to my dissertation jury members Assoc. Prof. Birol Aslanyürek and Prof. Dr. Samet Yücel Kadıoğlu for their time and valuable feedbacks. I would like to thank Professor Duc Truong Pham. He is supervised and encouraged me both as an independent researcher and as a team member. I am grateful for all his support, advice and contributions to my career and personal life. I would like to thank Dr. Yongjing Wang for his support and guidance when I first meet him at the University of Birmingham, Autonomous Remanufacturing (AUTOREMAN) Laboratory. Many thanks to all others AUTOREMAN team members Dr. Jun Huang, Mairi Kerin, Joey Lim, Mo Qu, Nathinee Theinnoi and others for their friendship and support. It was a pleasure to be a member of an interdisciplinary research group and to work together. I would like to special thanks to the members of Bees Algorithm Research Group; Dr. Asrul Harun Ismail, Natalia Hartono, Dr. Mario Caterino, Kaiwen Jiang, Dr. Turki Binbakır, and Dr. Murat Sahin for their friendship. I am fortunate enough to meet great people to work with. I would like to thank to Dean of Engineering Faculty, Prof. Dr. Ali Yılmaz Çamurcu, also thanks to Fatih Sultan Mehmet Vakif University (FSMVU) for their research support, and many thanks to the Dr. Berna Kiraz and members of the Department of Computer Engineering for the times we shared. I would like to thank my office-mate Derya Malkoç, and also many thanks to Merve Demir, Burçin Özbay, and Gönül Temiz for their friendships and supports. This research was supported by the Scientific and Technological Research Council of Turkey (TUBITAK), 2214-A International Research Fellowship Programme, Grant No. 1059B141800193. I would like to thank to the TUBITAK for their generous research support. But most of all, I want to express the deepest gratitude to my family; my dad Hüseyin Zeybek, my mom Ayşekadın Zeybek, my sister Melek Zeybek, and my brother Ramazan Zeybek for their love and unconditional support. Without their support, this thesis could not have been completed. Thank you.

Sultan ZEYBEK

TABLE OF CONTENTS

LIST OF SYMBOLS

φ	Activation Function
B	Bias Matrix
L_{BCE}	Binary Cross-Entropy Loss Function
L_{CE}	Cross-Entropy Loss Function
x^i	i^{th} input vector of the dataset
y^i	i^{th} Predicted Output of the Model
xp_k	k^{th} Parent Vector of the Parsed Tree
θ	Learnable Parameters of the Model
η	Learning Rate
hg_L	Left Hidden Child of the Inner Parse Tree
δ_i^k	Local Error Signal of Neuron i in Layer k
L_{MSE}	Mean Squared Error Loss Function
ngh	Neighbourhood Size
$(x^{(i)}, t^{(i)})$	One Sample Pair of the Dataset
$\nabla_\theta L$	Partial Derivative of the Loss function According to θ
$P(w\|c)$	Probability of Word w using Previous Context
hg_R	Right Hidden Child of the Inner Parse Tree
t^i	Target Data Label of the i^{th} input
V	Vocabulary
W	Weight Matrix
$w_{ij}^{(k)}$	Weight Vector from Neuron i to j Neuron at Layer k
x_w	Word Embedding of the Token w

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ACT-LSTM	Attentive Compositional Mechanisms in Binary Tree-LSTM
ANN	Artificial Neural Network
BA	Bees Algorithm
BA-3+	An Improved Ternary Bees Algorithm
BERT	Bidirectional Encoder Representation from Transformers
BP	Backpropagation
BPTT	Backpropagation Through Time
BPTS	Backpropagation Through Structure
BOW	Bag-of-words
CBOW	Continuous Bag-of-words
CNN	Convolutional Neural Networks
CT-LSTM	Constituency-Tree-Structured Long-Short-Term Memory Network
DL	Deep Learning
DNNs	Deep Neural Networks
DT-LSTMs	Dependency-Tree-Structured Long-Short-Term Memory Network
FFNN	Feedforward Neural Network
GA	Genetic Algorithm
GD	Gradient Descent
LSTM	Long-Short-Term Memory Network
ML	Machine Learning
MLP	Multilayer Perceptron
MRL	Morphologically Rich Language

MS-TR	Morphologically Enriched Turkish Sentiment Treebank
MSE	Mean Squared Error
NB	Naive Bayes
NER	Named Entity Recognition
NLP	Natural Language Processing
NMT	Neural Machine Translation
NN	Neural Network
OOV	Out-of-vocabulary
PCA	Principal Components Analysis
POS	Part-of-Speech
PSO	Particle Swarm Optimisation
QA	Question Answering
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Networks
RNTN	Recursive Neural Tensor Networks
SA	Sentiment Analysis
SG	Skip Gram
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SPINN	Stack-augmented Parser-Interpreter Neural Network
TF-IDF	Term Frequency-Inverse Document Frequency
Tree-LSTMs	Tree-Structured Long-Short-Term-Memory Networks
Tree-RNNs	Recursive Neural Networks
VEG	Vanishing and Exploding Gradients

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Recursive Deep Learning for Turkish Sentiment Analysis

Sultan ZEYBEK

Department of Mathematical Engineering
Doctor of Philosophy Thesis

Supervisor: Prof. Dr. Aydın SEÇER
Co-supervisor: Assist. Prof. Dr. Ebubekir KOÇ

In this thesis, Recursive Deep Learning models have been implemented for Turkish sentiment analysis. Although natural language processing has made progress recently, representing compositional meanings is a challenging task. The traditional deep learning methods claim sentences as an ordinary linear structure, i.e. chains or sequences. In this thesis, tree-structured representations of the language have been developed to improve the compositional semantics of the Turkish language considering the morphological structure of the words. To this end, a novel Morphologically Enriched Turkish Sentiment Treebank (MS-TR) has been constructed to encode sentences structure. MS-TR is the first fully-labelled sentiment analysis treebank, which has four different annotation levels, including morph-level, stem-level, token-level, and review-level. Recursive Neural Tensor Networks (RNTN), which operate over MS-TR, have achieved much better results compared to the machine learning methods. In addition to the RNTN model, an advanced tree-structured LSTM model (ACT-LSTM) has been proposed as a novel recursive deep architecture. ACT-LSTM combines both attention and memory mechanisms over recursive tree structures, which learn latent structural information while learning more important parts of the sentences. ACT-LSTM has been compared with advanced chain-structured models to decide which architecture is better. As a third main contribution, a novel metaheuristic training algorithm has been proposed to overcome the vanishing and exploding gradients (VEG) problem, which is usually observed while training models. An enhanced ternary Bees Algorithm (BA-3+) has been implemented, which maintains low time complexity for large dataset classification problems by considering only three individual solutions in each iteration. The algorithm utilises the greedy selection strategy of the local solutions

with exploitative search, stabilises the problem of VEGs of the decision parameters using SGD learning with singular value decomposition, and explores the random global solution with explorative search. BA-3+ has achieved faster convergence, avoiding getting trapped at local optima compared to the classical SGD training algorithm.

Keywords: Recursive deep learning, deep neural networks, sentiment analysis, natural language processing, Turkish sentiment analysis

ÖZET

Yinelemeli Derin Öğrenme Teknikleri ile Türkçe Duygu Analizi

Sultan ZEYBEK

Matematik Mühendisliği Anabilim Dalı
Doktora Tezi

Danışman: Prof. Dr. Aydın SEÇER
Eş-Danışman: Dr. Öğr. Üyesi Ebubekir KOÇ

Bu tez çalışmasında, yinelemeli (recursive) derin öğrenme mimarileri kullanılarak Türkçe'nin biçimbilimsel ve anlambilimsel özelliklerinin bileşkesel modellenmesi amaçlanmıştır. Önerilen yinelemeli derin öğrenme modelleri Türkçe duygu analizi görevi üzerinden test edilmiştir ve geleneksel zincir-yapılı (recurrent) modeller ve makine öğrenmesi modelleri karşılaştırılmıştır. Yinelemeli derin öğrenme mimarilerinin eğitilmesi için öncelikle ağaç-yapılı etiketlenmiş veri kümelerinin oluşturulması gerekmektedir. Bu bağlamda, kelimelerin morfolojik özellikleriyle kök + ekler, kök, kelime ve doküman seviyeleri olmak üzere dört farklı seviyede etiketlenmiş yeni bir duygu bankası (MS-TR) oluşturulmuştur. MS-TR gelecekteki Türkçe yinelemeli derin öğrenme modellerinde kullanılabilecek ilk etiketlenmiş duygu ağacı olması nedeniyle literatüre oldukça önemli bir katkıda bulunmuştur. Yinelemeli Tensör Sinir Ağı, MS-TR üzerinden ikili ve incetaneli duygu analizi görevi için cümle temsillerini modellemiştir. Yinelemeli Tensör Sinir Ağı modeli, SVM, NB ve MaxEnt gibi makine öğrenmesi modellerine kıyasla en yüksek başarı oranlarını göstermiştir. Yinelemeli Tensör Sinir Ağı modelinin performansını iyileştirmek için ACT-LSTM adlı yeni bir ağaç-yapılı sinir ağı modeli önerilmiştir. ACT-LSTM dikkat ve hafıza mekanizmalarını birleştirerek verinin daha önemli olan kısımlarına odaklanarak öğrenmeyi sağlamaktadır. Böylece kutupluluk içeren kelimeler ağırlıklandırılmış yorum temsiline daha fazla katkı sağlamaktadır. ACT-LSTM, diğer zincir-yapılı modeller ile karşılaştırılarak hangi mimarinin daha iyi olduğu araştırılmıştır. Tezin son kısmında eğitim sırasında sıklıkla ortaya çıkan kaybolan ve patlayan türev (vanishing and exploding gradient) problemine çözüm önerisi olarak yeni bir metasezgisel öğrenme

algoritması önerilmiştir. Bu amaçla, her bir arının ayrı bir öğrenme modelini temsil ettiği geliştirilmiş üçlü arı algoritması (BA-3+) önerilmiştir. BA-3+, stokastik gradyan inişi ve tekil değer ayrıştırma mekanızmasını birleştirerek kaybolan ve patlayan türev problemi nedeniyle bilginin kaybolmasını önler. Lokal arama, stokastik gradyan inişi ve tekil değer ayrıştırma kombinasyonu ile en iyi parametre değerlerine yakınsamayı garantilerken, aynı zamanda global arama ile lokal optimum değerine takılıp kalmayı önlemektedir. Önerilen eğitim algoritması, yalnızca stokastik gradyan inişi kullanan diğer modellere kıyasla Türkçe duygu analizi için çok daha iyi doğruluk sonuçları elde etmiştir.

Anahtar Kelimeler: Yinelemeli derin öğrenme, derin sinir ağları, duygu analizi, doğal dil işleme, Türkçe duygu analizi

1 INTRODUCTION

1.1 Literature Review

Recently, the development of social media platforms has allowed people to express their actions, their beliefs, and their lives, employing figurative and creative language. Constructing different data sources from social media, movie ratings, and product reviews offer a strategical understanding of human behaviours and thoughts as the rapid growth of Web 2.0 has increased Internet user data.

Artificial intelligence (AI) is a research topic, which aims to mathematically model the human thinking system and develop intelligent systems that can think and learn like humans. Natural language processing (NLP), as a sub-field of AI, enables language processing for specific purposes such as information extraction, meaning-extraction and sentiment analysis. Sentiment analysis (SA) is a prevalent research topic concerned with classifying and exploring the views in a specific text, document, or sentence. SA studies have a significant potential effect on from academia to commercial applications to get an insight into peoples opinions as a crucial factor that influences human behaviours.

Learning the sentence structure and understanding the meaning of them in a computer is a challenging task. The main difficulty in terms of semantics is to capture the meaning of longer phrases. Even though NLP methods have made progress in recent years, representing compositional meanings is a challenging field of NLP. Current machine learning algorithms have certain drawbacks, such as limitations of generalising when applied unstructured text data. For example, the traditional feature representation methods such as bag-of-words (BOWs) represents words as a one-hot vector, which causes the loss of the semantic relationships between words due to the atomic representations. Besides sentences could be represented as high-dimensional sparse encodings based on the word counts in a large vocabulary, which are not well enough to represent the sophisticated, compositional semantics of a language. Hence much more studies should have been done to fill the gap in compositional representation learning.

So far, most of the sentiment analysis and compositional representation learning studies have focused on English. The first study was done by Pang *et. al* [1] suggested using the BOW features and classifying reviews using Support Vector Machine (SVM) and Naive Bayes (NB). This study has been followed by most machine-learning-based approaches as a baseline study, and many other ML methods have been developed to learn dataset over features, such as n-grams, TF-IDF, part-of-speech (POS) tags, and polarity (sentiment) lexicons.

Linguists state that although the number of words in a natural language is limited, an infinite number of sentences can be created by the natural language speakers. Grammar rules are used as a natural way to create languages. Determining the compositional meaning of the language (i.e. distributed representations for phrases, clauses and sentences) is very hard since the language is represented as atomic, discrete values and symbolic data structures which are unable to keep dynamical flows of the information. Both one-hot-encodings and word embeddings have limited ability to model connectional meanings of at least two items, and they cannot capture the semantic information of given sentences since they cannot combine structurally or sequentially.

Human thinking and speaking happen in a non-linear way, and language is produced *recursively* by interrelated processes in mind as a result of the integration and interpretation ability of the human mind. It is crucial to represent this structural information for understanding the compositional semantics of the natural language, which is processed between the human mind and language recursively. The traditional methods claim sentences as an ordinary linear structure, i.e. chains or sequences. However, finding the optimum way to detect the connectionist structure of the data and representing it in an optimum structure is still an open problem.

Deep learning provides learning high-level and abstract features automatically by increasing hidden layers and gives better representations of data that makes training easier in the sense of optimisation procedure of NLP task. Through deep layers, the task can be done without time-killer feature engineering steps of traditional machine learning. The use of better representations, i.e. dense word embedding also be an efficient way for dealing with high dimensions, which appears as a result of the representing words like atomic units in a one-hot vector space that ignoring word order. With the *deep learning* era, *Neural Language Models (NLM)* get dramatic successes in semantics and sentiment analysis tasks with the contribution of the large data sets and improvements. The magic behind the success of all of the learning models is dependent on how well data is represented without losing information. Recursive Neural Networks are developed for sentiment analysis as they can capture the semantic compositionality in a given sentence [2, 3]. The standard version of the Recursive Neural Networks (Tree-RNNs), namely Recursive Autoencoders

(RAE), was introduced by Socher *et al.* [4] using autoencoders to understand language as a continuous architecture of meaning, which flows from characters to words, words to phrases, and phrases to sentences in a recursive manner. A semi-supervised version of the RAE can learn phrase representations to predict sentiment distributions of the sentences with good accuracy levels; however, it is not able to capture long-phrase meanings and all types of compositions.

Researches focusing on compositional models for morphologically rich languages (MRLs) is still developing, and they have not gone far beyond the conventional approaches. However, these approaches have a performance bottleneck due to representing words as independent atomic units, which causes the loss of explicit information among words and reducing accuracy. Hence new language representation models are needed to be developed to learn the nonlinear structures of languages. However, fewer studies are done for additive structured languages such as Turkish, and success rates are lower than in English studies. Moreover, since different prepossessing techniques are required for each language, the tools developed for English cannot be directly used in languages with rich morphology, such as Turkish.

Turkish is a morphologically rich language (MRL). Many different words can be derived by adding various suffixes to each word root. Hence the number of different words increases, the space size becomes too large, and the processing time takes too much time. In addition, since the data to be used during emotion analysis is raw data, many noise and spelling errors in the raw data are corrected during the preliminary preparation stage on the data set. However, this may cause some expressions that express emotion to be lost. When using the commonly used word bag model, one has to work with large areas.

Feature engineering within the scope of machine learning is closely concerned with feature extraction from raw data. Current Turkish SA studies have mainly focused on machine-learning-based approaches and lexicon-based approaches, trying to learn the dataset features such as unigram, bigram, Part-of-speech (POS) tags or predict the emotional orientation of an input text by using the emotional score of words. However, these features are not enough to learn higher-level nonlinear features from large amounts of data, although designing them take too much time.

Erogul's thesis is the earliest Turkish SA work that was done in 2009 [5]. He built the tagged movie review dataset from beyazperde.com and classified them by using SVM. He investigated the effect of the stemming process and part-of-speech (POS) tags on the Turkish sentiment model performance. As a baseline, he used bag-of-words features with spellchecking and elimination process and achieved 85% F1-score. As a special case, he focused on the different features, including using only roots of the words, parts-of-speech

(POS) tags, n-grams. Using only the roots of the words as a feature decreased performance to the 83.99% F1-score. Noun + Adjective features performed better compared to the other part-of-speech tags features and achieved 83.34% F1-score. The best accuracy had been achieved as 86.16% F1-score, which has obtained by the combination of the unigram, bigram, trigram and 4 gram.

One other remarkable study was done to classify Turkish political news by Kaya *et al.* [6]. They used a combination of n-grams, root words, adjectives, and polar words feature to represent political news in the bag-of-words (BOW) framework. Maximum Entropy and n-gram LM achieved better accuracy levels compared to the SVM and NB with the 76-77% accuracy level.

A model had been proposed for assigning polarities of the Turkish blog posts about products and services by Aytekin [7], that uses Naïve Bayes Algorithm to classify 350 positive and 350 negative comments which were collected by the We Feel Fine website. A sentiment dictionary, including 4,744 synsets, was used to assign probability distributions. The precision scores measured up to 72.28% (pos) and 73.14% (neg), respectively.

Even though machine-learning algorithms have reached a good success rate, lexicon-based approaches are widely preferred because they are practical. They are simply applied by attaching a sentiment polarity score to the words or phrases based on the lexicon information. As an earlier lexicon-based work, Vural *et al.* proposed to use SentiStrenght as a polarity lexicon [8] to classify movie review dataset with unsupervised learning. They translated the SentiStrenght's polar words from English to Turkish [9] and calculated the total polarity score of the original input text to detect positive and negative polarity score. In terms of the accuracy result, their framework has reached up to 75.90% for the binary classification task.

Over the last years, some additional efforts have been made to built Turkish polarity lexicons. Dehkharghani *et al.* built a polarity lexicon, namely SentiTurknet, which contains 14,795 synsets each has three-level (positive, negative, neutral) polarity score [10]. The polarity scores evaluated by complementary usage of various NLP resources such as English Wordnet [11], Turkish Wordnet [12] lexicons and English SentiWordNet [13], English SenticNet [14, 15] and Polar Word Set (PWS) and polarity words with Pairwise Mutual Information (PMI) polarity scores. The proposed algorithm combined manual labelling and feature extraction steps to detect the polarity of a given synset. Additional polarity features were extracted from polarity lexicons, and they classified by logistic regression (LR), neural networks and SVM. The final estimation of the label was predicted by the combination of these three classifiers. SentiTurkNet was tested on the Turkish movie reviews from beyazperde.com and achieved between 61.3% - 66.7% accuracy for

ternary sentiment classification.

Beyond the above, BOW representation and n-grams have become the most preferred feature modelling methods. Coban *et al.* [16] constructed a Twitter dataset for binary sentiment classification, and they used BOW and N-grams to extract features. For all case studies, N-gram features performed better compared to the BOW and achieved between 62%- 66% accuracy level. Extreme Learning Machine (ELM) was used to classify tweets, and customer reviews of the telecom company based on the BOW and N-gram features compared to SVM. SVM performed better with the highest accuracy level 74%, which is not good enough again for the datasets, which each has lower than 3000 entries [17].

Few studies have been taking into account the morphological features of Turkish to detect sentiment. Several NLP modules have been demonstrated on the noisy dataset in [18]. The study proposed to use normalisation, negation handling, morphological analysis and stemming modules and also adjectives for ternary classification using SVM. Experiments achieved up to 79% accuracy level with the usage of normalisation and stemming features, which means that additional morphological information improved the system performance. Similarly, Turkmenoglu and Tantug [19] suggested comparing ML and lexicon-based approaches taking into account morphological features of Turkish, i.e. taking into account absence/presence suffixes to detect kind of negation in Turkish words. The SentiStreght was used as a baseline lexicon. Additionally, the booster list was used to detect adjectives polarity score. The machine learning approach followed a similar way to the studies above, and TF-IDF was used with unigrams and bigrams features. As a classification algorithm, SVM, Naïve Bayes and Decision Trees were tested by using Twitter and Movie dataset. The lexicon-based approach combined with all modules, including normalisation, negation handling, multi-word expressions and booster word list, achieved 75.2% and 79% and accuracy level for Twitter and Movie dataset, respectively. Machine learning approaches reached up to 85% accuracy level by using TF-IDF and unigram and bigram features of tweets classified by SVM, and SVM and Naïve Bayes achieved 89.5% accuracy level on Movie dataset with same features.

As an extended version of the Turkmenoglu and Tantug's lexicon-based approach, Yurtalan *et al.* [20] proposed to used conjunctions to detect opposite meanings between two phrases, idioms and proverbs analyzer and multi-word analyzer in addition to the negation handling phase of the word-level analyzer using 1181 polar items. They scored each word with the polarity value ranging from -3 to +3 and classify each review according to the total polarity value. Some researchers have been focused on constructing datasets and annotation tools for sentiment analysis. TURKSENT had been implemented as an annotation interface particularly for sentiment analysis from social media [21] by combining linguistic annotation layer including morphological analysis and normalization features,

and human annotation layer.

Despite all these time-consuming feature engineering efforts, the accuracy level of the proposed models are still not good enough, and it is necessary to improve more efficient ways. Notably, for the Turkish language, many methodologies which are successfully applied to the English language are waiting to be explored to handle the challenging nature of Turkish sentiment analysis.

1.2 Objective of the Thesis

The research aim is to develop tree-structured (recursive) deep learning models to learn compositional representations of morphologically enriched words for Turkish sentiment analysis.

The research objectives are as follows:

- To develop tree-structured dataset and recursive deep learning models for capturing longer word dependencies and compositional representation of the Turkish language.
- To prevent the loss of information caused by long-term time and long-term tree dependencies by combining the attention and memory mechanism of the novel recursive model, which is proposed as ACT-LSTMs.
- To improve the training algorithm of the architectures for avoiding vanishing and exploding gradients (VEG) problem that occurs while learning deep recurrent and recursive layers by employing an enhanced ternary Bees Algorithm (BA-3+) as a novel meta-heuristics learning method performing local learning with exploitative search, SGD learning with singular value decomposition (SVD), and global learning with the explorative search.

1.3 Original Contribution

The original contributions of this thesis are:

- A novel Morphologically Enriched Turkish Sentiment Treebank (MS-TR) was constructed to address compositional sentiment analysis for Turkish by using Recursive Deep Models.
- A novel semi-supervised automatic annotation was proposed as a distant-supervision approach. The morphological features of words were used to infer the

polarity of the inner nodes of the binary and fine-grained parse trees of the MS-TR. To the best of our knowledge, MS-TR is the first sentiment treebank that is fully labelled according to morphological structures of Turkish words. The effect of using morphological features of words on the performance was investigated using four different annotation level, including morph-level annotation, stem-level annotation, token-level annotation, and review-level annotation.

- Tree-structured Recursive Neural Tensor Networks (RNTN) were operated over MS-TR for Turkish Sentiment analysis, and better performances were obtained compared to the conventional machine learning methods, which combined various feature representation methods.

- Attentive composition mechanisms and memory blocks over recursive structures (ACT-LSTMs) were proposed as a novel tree-structured deep learning model to overcome long-term-dependencies and padding issues while improving structural learning for longer phrases. LSTM architecture had been generalized to process parsed trees while preventing the right-to-left reading of the human mind while preserving the linguistic structure of the sentences with attention.

- The comprehensive benchmark was done to compare the performances of advanced chain-structured and tree-structured language models to decide which architecture is better.

- A novel metaheuristic learning algorithm is developed to overcome the vanishing and exploding gradients problem of the deep learning models. An enhanced ternary Bees Algorithm (BA-3+) was proposed as a novel metaheuristic optimisation approach, which combines the collaborative search of three bees, performing local learning with exploitative search, Stochastic Gradient Descent (SGD) learning with singular value decomposition (SVD) to overcome vanishing and exploding gradients problem of deep architectures, and global learning with the explorative search for avoiding getting trapped at local optima.

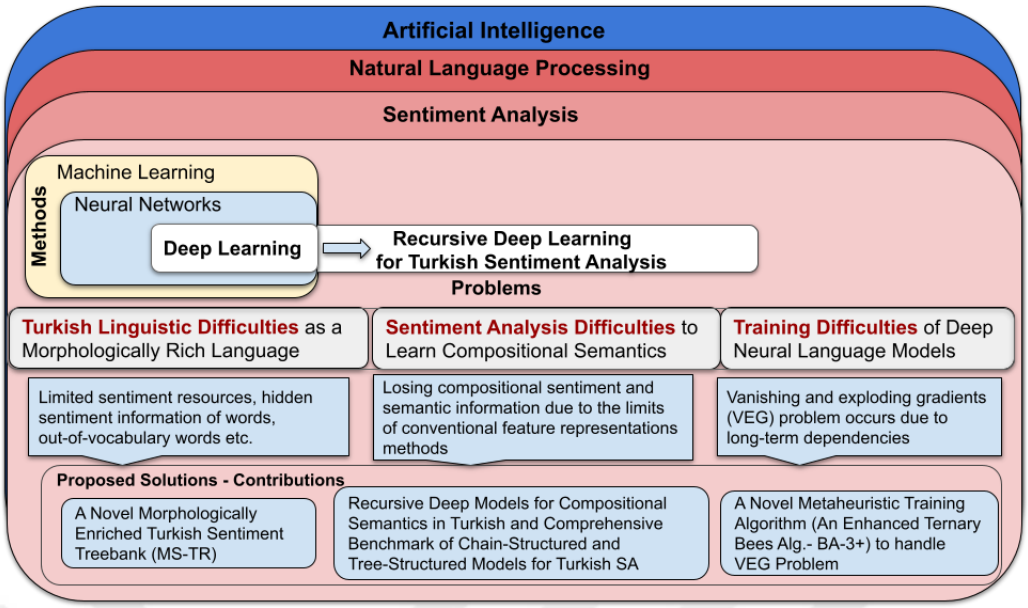

Figure 1.1 Proposed solutions and contributions of the thesis

1.4 Outline of the Thesis

Chapter 1 presents the aim and contributions of this study, problem definition, levels-resources of sentiment analysis and related work.

Chapter 2 covers the mathematical background to understand the recursive deep learning models methodologically for the Turkish sentiment analysis task. It reviews the basic notations, definitions, and methods for deep neural network-based language models. The theory of classification is given for a better understanding of sentiment classification task. Distributional representations, gradient-based training algorithms and training difficulties of deep models such as vanishing and exploding gradients problem are presented from the perspective of mathematical optimisation. Terminology and difficulties of the sentiment analysis have been also given. Finally, the characteristic and challenges of Turkish language have been discussed in terms of sentiment analysis.

Chapter 3 presents a novel Morphologically Enriched Sentiment Treebank for Compositional Semantics in Turkish (MS-TR). MS-TR is the first fully labelled according to morphological structures of the words to infer the polarity of the inner nodes of MS-TR as positive and negative. MS-TR contributes to the lack of sentiment analysis resources in Turkish as an obligatory input format to work with Recursive Deep Models. A semi-supervised annotation model that has been done for four different levels, including morph-level, stem-level, token-level, and review-level, has been proposed. Morph-Level annotated MS-TR is aiming to retrieve the hidden polarity of the suffixes in the morphologically rich word. To this end, tokens are morphologically analyzed, and they are parsed to their possible stem and suffixes. Using only a morphological analysis of the word can-

not provide the correct polarity information for each case. Hence, in addition to the morphological annotation, using a polarity lexicon was proposed to capture polar words, which are root and do not have any morphological information. Stem-Level annotated MS-TR was constructed similar to morph-level annotated MS-TR. The only difference is using only stems of the words, and the suffixes (ending) of the words are eliminated from the word after morphological parsing. As a third level, each token of the reviews is annotated using polar embedding spaces, which are constructed by using positive and negative datasets. The cosine similarity measure has been used to find the token-level label. As the last annotation level, only review-level annotated tree structures have been used to construct MS-TR for comparing each annotation level's performance. In addition to the binary-labelled parse trees, the fine-grained dataset has been annotated by taking into account the morphological information of the words. The polarity distribution of the phrases, i.e. phrase-level to review-level labelling, was realized in two stages. The first stage is labelling words according to their polarity features which are detected from the morphological analysis of the words. In addition to the binary annotation, words have been scored whether they are booster words. Three different domain datasets were used, including product reviews, movie reviews, and the Turkish Natural Corpus essays for construction. Comparative results were obtained with the Recursive Neural Tensor Networks (RNTNs), which is operated over MS-TR, and conventional ML methods. Experiments proved that RNTN outperformed the baseline methods and achieved much better accuracy results compare to the baseline methods, which cannot accurately capture the aggregated sentiment information.

Chapter 4 focuses on improving recursive deep learning models using memory and attention mechanisms to learn latent structural information and for improving the performance of the Turkish Sentiment Analysis system. To this end, ACT-LSTMs, namely an attentive compositional mechanism in binary Tree-LSTMs, have been proposed to combine both attention and memory over recursive tree structures. LSTM model has been extended to ACT-LSTM, which can work over sentiment treebank while learning important (more related) part of the long sentences. The main motivation is using attention over tree structures and mimicking the human attention mechanism to memorize and learn the more important parts of the given sentences for a downstream task. The aim is to prevent the loss of information, which occurs due to padding operations. The performances of advanced chain-structured and tree-structured language models have been compared over a SA task to decide which architecture is better. According to the experimental results, ACT-LSTM performed better in terms of fine-grained Sentiment Analysis (SA). They combined the advantages of the attention mechanism with the combined power of the memory blocks, particularly to improve the performance of the structured fine-grained SA models. It had been observed that Tree-LSTMs performed better compared to chain-structured LSTMs

and RNTN. RNTN had also performed better than chain-LSTMs combined with various segmentation methods.

Chapter 5 presents a novel metaheuristic optimization approach to solve vanishing and exploding gradients (VEG) problem, as is discussed in detail in section 2.4.1. The approach employs an enhanced ternary Bees Algorithm (BA-3+) which maintains low time-complexity for large dataset classification problems by considering only three individual solutions in each iteration. The algorithm combines the collaborative search of three bees, performing local learning with exploitative search, SGD learning with singular value decomposition (SVD), and global learning with explorative search. Thus, the algorithm utilizes the greedy selection strategy of the local search operators of the basic Bees Algorithm to improve solutions, the stabilization strategy of SVD to handle the problem of VEG of the decision parameters, and the random global search strategy of the basic Bees Algorithm to achieve faster convergence avoiding getting trapped at local optima. BA-3+ has been used to find the optimal set of trainable parameters of the proposed deep recurrent learning architecture. The proposed algorithm has been compared for sentiment detection. According to the experimental results, the improved accuracy and convergence results showed that the proposed algorithm performed better compared to traditional SGD and BA-3+ is an efficient algorithm for training deep RNNs for complex classification tasks.

Chapter 6 reports results and discussion of the thesis. Suggestions are also given for further future researches.

2
BACKGROUND

2.1 Supervised Learning

The "analysis" part of the sentiment analysis of all levels is a classification problem in essence. Classification and deciding where the data belongs is the most crucial step in the machine learning mechanism. Conventionally most of the sentiment classification problems have been solved using *supervised learning*. This thesis aims to detect the sentiment orientation of a given expression that includes an opinion about a specified topic. More specifically, if we talked about the *binary-classifier* then the problem can be formulated as follows:

$$\zeta : X_{inputs} \rightarrow \ \mathrm{T} = \{0,1\} \tag{2.1}$$

Here the map ζ as a positive or negative predictor of the observations (i.e. inputs) of the corresponding class $T = \{0,1\}$. In most of the classifying cases, ζ has been learned via *supervised learning* over training set (D). The pairs $(x_i,\ t_i), 1 \leq i \leq n$ of D which are distributed randomly and the classes $t_i \in C$ have been given by instructor. The aim is learning ζ which can predict the actual (target) class, i.e. label t_i of observation x_i with minimal probability of loss.

The classifier success can be measured by the probability of loss function, which is defined as follows:

$$\mathrm{L}_\zeta = \mathrm{P}\big\{\zeta(X_{inputs}) \neq T \mid D\big\} \tag{2.2}$$

where $D = \big\{(x_i,\ t_i) \in X \times C, 1 \leq i \leq n\big\}$.

The aim is to learn the most suitable function over $\zeta \in \mathcal{F}$, which can as much as minimize the disagreement between the predicted class and the actual (gold) class. In the case of sentiment analysis, the sentiment class of x_i, i.e. sentiment observations of each review

or tweet from the dataset is predicted by classifier function, $\zeta : x_i \rightarrow y_i \in \{0,1\}$, one for positive sentiment class and zero for negative sentiment. Each pair of $(y_i,\ t_i)$ has been used to maximise the correct estimation or minimise the *error* between predicted y_i and target data t_i. This process defines our *objective function (loss function)* throughout this thesis as follows:

$$\mathrm{L}_{\zeta}(y_i,\ t_i) = \mathrm{P}\big\{\zeta(X_{inputs}) \neq T \mid D\big\} \tag{2.3}$$

or for any $\zeta \in \mathcal{F}$ it is expected loss which is defined as follows:

$$\begin{aligned} \mathrm{L}_{\zeta}{}^{*} &= \underset{\zeta \in \mathcal{F}}{\operatorname{argmin}}\, \mathbb{E}_{(x,\ t)\sim D}\, \mathrm{L}(\zeta(x_i),\ t_i) \\ &= \underset{\zeta \in \mathcal{F}}{\operatorname{argmin}}\, \mathbb{E}_{(x,\ t)\sim D}\, \mathrm{L}(y_i,\ t_i) \end{aligned} \tag{2.4}$$

Since D represents the unknown distributions of data pairs (x,t) and the expected value of $\mathbb{E}_{(x_i,\ t_i)\sim D}$ could not precisely be known. Besides, since the pairs are distributed identical and independent (i.i.d) over D, L_{ζ} can be referred using *average of total training error* represented by $\mathrm{L}_{\zeta}{}^{*}$:

$$\mathrm{L}_{\zeta}{}^{*} \approx \frac{1}{N}\sum_{i=1}^{N} \mathrm{L}_{\zeta}(y_i,\ t_i) \tag{2.5}$$

2.2 Deep Learning

2.2.1 Neural Networks

A basic neuron is a simple summation function, which is designed years ago [22] in 1943. This mechanism is a mathematical model of the biological neuron. After a while, Hebb [23] reported enhancement of human learning by interconnections between neurons.

Let $\mathrm{X} = (x_1, x_2, \ldots, x_k)$ be a k-dimensional input vector. A single neuron is a mapping that

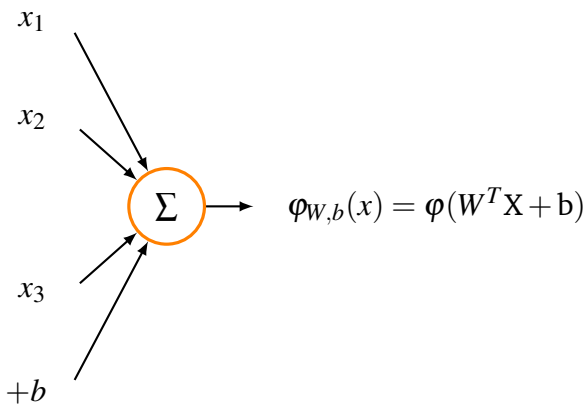

Figure 2.1 A single neuron example with 3 inputs, bias and 1 output

can learn weights through following summation:

$$\begin{aligned} z^{(0)} &= \sum_{i=1}^{d} {w_i}^T x_i + b \\ a^{(0)} &= \varphi(z^{(0)}) \\ &= \varphi(W^T \mathrm{X} + \mathrm{b}) \end{aligned} \tag{2.6}$$

Here φ is an "*activation function*", which transforms the nonlinear features of the input values for a specified task.

$$y = \begin{cases} 1, & \text{if } \varphi(\sum_{i=1}^{d} {w_i}^T x_i + b) \geq 0, \\ 0, & \text{otherwise,} \end{cases}$$

Accordingly, Rosenblatt proposed a remarkable *perceptron learning* rule as a way for updating the weights of the ANN system:

$$w_i^{\text{new}} = w_i^{\text{old}} - \eta \cdot (y_i - t_i) \cdot x_i \tag{2.7}$$

to minimize the expecting error between $(y_i,\ t_i)$ as given in Equation 2.4. Here η is a critical *hyper-parameter* of the model allowing adjustments to which weight to update. Learnable parameters of the neuron in Figure 2.1 are weights (W) and single bias (b) value. Although this can be a practical binary classifier, perceptron's hyper-plane cannot classify the non-linear dataset (see Figure 2.2).

Figure 2.2 represents an example for distributed data point with a different separability nature. The data points of (i) can be separated using linear mapping, and (ii) can be

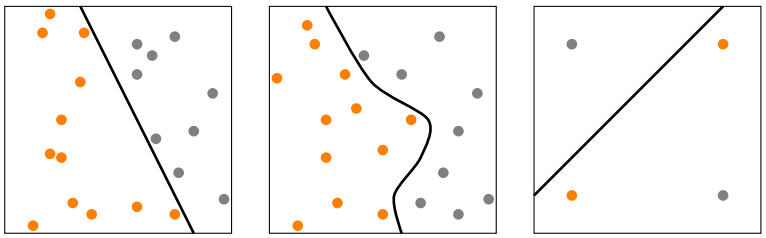

Figure 2.2 Examples of datasets with different separability

separated using a nonlinear function. However, (iii) (XOR problem) cannot be separated with a simple mapping. Models need to handle the nonlinear nature of the data. With this motivation, Multilayer neural architectures (i.e. MLP) have been implemented [24] which is combined with middle layers and m units for each (see Figure 2.3).

Algorithm 1: Deep Layer Feedforward NN

Input: k = hidden layers , input vector=x,

$weight\ matrices = \{W_0, W_2, \ldots, W_k\}$, $biases = \{b_1, b_2, \ldots, b_{k+1}\}$

$h_0 \leftarrow x$

for $i \leftarrow 1$ ***to*** $k+1$ **do**

  $x_i \leftarrow h_{i-1} W_{i-1} + b_i$

  $z_i \leftarrow \alpha_i(x_i)$

return $y \leftarrow z_k$

A deep feedforward NN (FFNN) is a mathematical processing unit, which is defined with input layers, hidden layers and outputs layers as showed in Figure 2.3. It performs an activation functions considering the weighted summation of input vectors and bias scalar to learn nonlinear features of the given inputs. Algorithm 1 explains the steps of simplest version of the deep feedforward neural networks with an activation function α, which can be selected as sigmoid, tanh, ReLU functions. The algorithm is the pseudocode of the forward pass for one sample $(x^{(i)}, t^{(i)})$ pair from the dataset. φ the nonlinear activation function, and the total loss function calculated from the summation of each pair loss.

Activation of the hidden layers for a single neuron is calculated in a similar way with Equation 2.6 as follows:

$$\begin{aligned} z^{(k)} &= {W_{ih}}^k \, a^{(k-1)} + {b_{ih}}^k \\ a^{(k)} &= \varphi(z^k) \end{aligned} \tag{2.8}$$

where it is called as *forward propagation* chain. Here W_{ih}, b_{ih} are learnable parameters of the MLP model given in 2.8, where the weight matrix $W_{ih} \in \mathbb{R}^{kxn}$ since it is assumed that

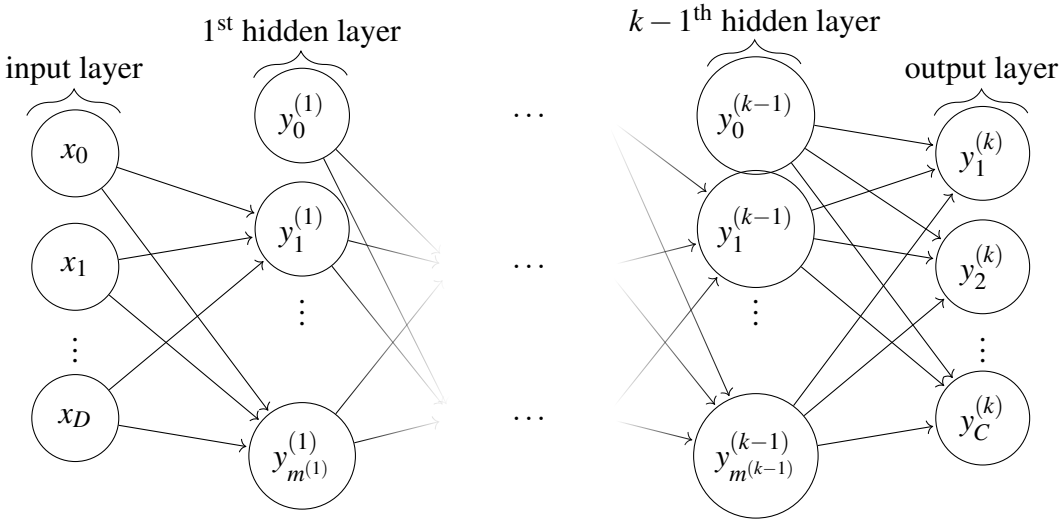

Figure 2.3 A MLP with hidden layers

MLP model has k hidden layer. Throughout this thesis the following nonlinear functions have been used:

$$\varphi = \begin{cases} \text{sigmoid}, & \frac{1}{1+\exp(-z^k)} \\ \tanh, & \frac{1-exp(-2z^k)}{1+exp(-2z^k)} \\ \text{softmax}, & \frac{\exp(z^k)}{\sum_{i=0}^{C}\exp(z^i)} \end{cases} \tag{2.9}$$

Here W_{ih} represents the weight matrix from hidden layer $(k-1)$ to hidden layer k, and C is the number of classes. It can be seen that clearly, the outputs are represented as any combination of the inner hidden layer activation, which is mapped into non-linear features with activation functions that are chosen from Equation 2.9.

This mechanism opened a new era inspiring the different kinds of deep learning models. They are designed with "*hidden layers*" as an automatic feature learners. They can solve many tasks while exploring the nature of the data without hand-generated feature vectors.

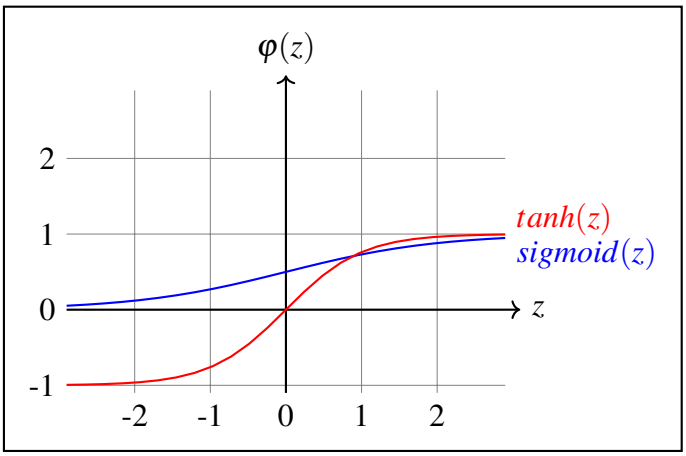

Figure 2.4 Examples of activation functions

Activation functions are critical mappings for modelling the nonlinearity of inputs. There are many commonly used functions. Figure 2.4 represents the activation functions of sigmoid and tanh, and the two most common of these. MLPs combine affine mappings and nonlinearity to learn the features of the problem for a specified task. It is good to remind that, without nonlinearity, it is impossible to learn the features of data, and the MLP system would be the same as simple linear mapping. Besides the advantages of the nonlinear affine transformation, selecting the most suitable activation function is still an open research topic due to the training difficulties of the deep architectures, which is discussed in section 2.4 in detail.

2.2.2 Training: Backpropagation

Training of a learning model aims to converge the optimum values of the learnable parameters of the model. The first version of the learning rule for neural models is introduced in Equation 2.7 as perceptron learning. Perceptron algorithm cannot be applied directly to the MLP models since the hidden layers don't have target values.

Gradient Descent (GD) is an algorithm to update multi-layer models parameters using following equation:

$$\theta_t = \theta_{t-1} - \eta \left. \frac{\partial \mathrm{L}_\zeta}{\partial \theta} \right|_{t-1} \tag{2.10}$$

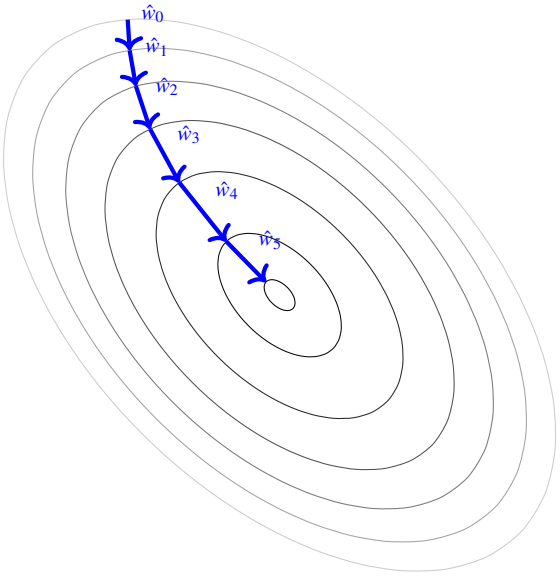

Figure 2.5 Gradient descent algorithm

which can also be represented as following:

$$\theta_t = \theta_{t-1} - \eta \, \nabla_\theta \, L(\varphi(x_i, \theta), t_i)) \tag{2.11}$$

where learning rate η is a crucial the hyper-parameter that defines the GD steps and helps to converge to optimum values of the parameters. Figure 2.5 illustrates the iterations of the gradient descent algorithm to converge optimum values of a given function. *Backpropagation (BP)* is a learning way to optimize the multilayer model parameters combining gradient descent and chain-rule. It plays a critical role to converge the optimum values of the learning parameters of the model. First chain-rule has been used to calculate the partial derivatives according to the specified parameter as follows:

$$\begin{aligned}
\frac{\partial \mathrm{L}\zeta}{\partial W_{ij}^{(k)}} &= \frac{\partial \mathrm{L}\zeta}{\partial z_j^{(k)}} \frac{\partial z_j^{(k)}}{\partial W_{ij}^{(k)}} = \frac{\partial \mathrm{L}\zeta}{\partial a_j^{(k)}} \frac{\partial a_j^{(k)}}{\partial z_j^{(k)}} \frac{\partial z_j^{(k)}}{\partial W_{ij}^{(k)}} \\
\frac{\partial z_j^{(k)}}{\partial W_{ij}^{(k)}} &= \frac{\partial \sum_i W_{ij}^{(k)} x_i^{(k-1)}}{\partial W_{ij}^{(k)}} = a_i^{(k-1)} \\
\frac{\partial a_j^{(k)}}{\partial z_j^{(k)}} &= \frac{\partial \varphi(z_j^{(k)})}{\partial z_j^{(k)}} = (\varphi(z_j^{(k)})(1-\varphi(z_j^{(k)})) = o_j(1-o_j) \\
\frac{\partial \mathrm{L}\zeta}{\partial a_j^{(k)}} &= \frac{\partial \mathrm{L}\zeta}{\partial o_j} = \frac{\partial \sum_{i=1}^{k} (t_i - o_i)^2}{\partial o_j} = (o_j - t_j) \\
\frac{\partial \mathrm{L}\zeta}{\partial W_{ij}{}^{(k)}} &= (o_j - t_j) o_j (1-o_j) a_i{}^{(k-1)}
\end{aligned} \tag{2.12}$$

Now the error for neuron j can be defined as following:

$$\begin{aligned}
\delta_j^{(k)} &= \frac{\partial \mathrm{L}\zeta}{z_j^{(k)}} \\
&= \frac{\partial \mathrm{L}\zeta}{a_j^{(k)}} \frac{\partial a_j^{(k)}}{z_j^{(k)}} \\
&= (o_j - t_j) o_j (1-o_j)
\end{aligned} \tag{2.13}$$

and the learnable parameters derivative is defined as follows:

$$\frac{\partial \mathrm{L}\zeta}{\partial W_{ij}^{(k)}} = \delta W_{ij}^{(k)} \tag{2.14}$$

where δ is defined as error message as illustrated in Figure 2.6.

Let the $W_{ih}^{(1)}$ represents the 1^{st} layer weight parameter. Chain-rule is allowed to calculate

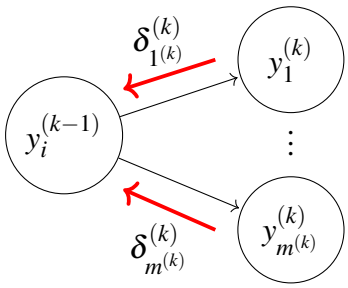

Figure 2.6 Backpropagation of errors through the network. $\delta_i{}^k$ represents the local error signal of neuron i in layer k.

the partial derivative of the output as follows:

$$\begin{aligned}
\frac{\partial y}{\partial W^{(1)}_{ih}} &= \frac{\partial y}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial W^{(1)}_{ih}} \\
&= \frac{\partial y}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial W^{(1)}_{ih}} \\
&= \frac{\partial y}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial W^{(1)}_{ih}} \\
&= \frac{\partial y}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial a^{(1)}} \cdot \frac{\partial a^{(1)}}{\partial W^{(1)}_{ih}} \\
&= \frac{\partial y}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial a^{(1)}} \cdot \frac{\partial a^{(1)}}{\partial z^{(1)}} \cdot \frac{\partial z^{(1)}}{\partial W^{(1)}_{ih}} \\
&= \frac{\partial y}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial a^{(1)}} \cdot \frac{\partial a^{(1)}}{\partial z^{(1)}} \cdot \frac{\partial z^{(1)}}{\partial W^{(1)}_{ih}}
\end{aligned} \tag{2.15}$$

If $y = a^{(3)}$ is claimed as it is defined in 2.8:

$$\frac{\partial y}{\partial w^{(1)}_{ih}} = \frac{\partial a^{(3)}}{\partial z^{(3)}} \cdot {W_{ih}}^{(3)} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot {W_{ih}}^{(2)} \cdot \frac{\partial a^{(1)}}{\partial z^{(1)}} \cdot [a^{(0)}_j]_i \tag{2.16}$$

with $[a^{(0)}_j]_i$, which is a vector of the same size as $z^{(1)}$ whose ith element is equal to $a^{(0)}_j$ and all the other elements are 0. Each derivative $\frac{\partial a^{(k)}}{\partial z^{(k)}}$ corresponds to a $m \times m$ matrix (where

m is the number of unit in a one single layer) as follows:

$$\frac{\partial a^{(k)}}{\partial z^{(k)}} = \begin{bmatrix} \frac{\partial \varphi_0(z^{(k)})}{\partial z_0^{(k)}} & \cdots & \frac{\partial \varphi_0(z^{(k)})}{\partial z_m^{(k)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \varphi_m(z^{(k)})}{\partial z_0^{(k)}} & \cdots & \frac{\partial \varphi_m(z^{(k)})}{\partial z_m^{(k)}} \end{bmatrix} \tag{2.17}$$

equals to the following *diagonal matrix*:

$$\frac{\partial a^{(k)}}{\partial z^{(k)}} = \begin{bmatrix} \varphi'(z_0^{(k)}) & & 0 \\ & \ddots & \\ 0 & & \varphi'(z_m^{(k)}) \end{bmatrix} \tag{2.18}$$

The *stochastic gradient descent (SGD)* is a well-known generalised version of GD, which learn from one example pair at each iteration instead of using the entire dataset at each step. SGD particularly useful method for large training datasets. The gradient-based training algorithms use the backpropagation algorithm, which is calculating the partial derivatives according to learnable parameters of the system via the chain rule as given in Algorithm 2.

Algorithm 2: Stochastic gradient descent (SGD) algorithm

Input: Learning rate:η, $\theta = (W_{ih}, b_{ih})$
Output: θ^{t+1}
Function `SGD(`η, θ`)`**:**
  for *each hidden layer k* **do**
    Select training pairs $(x^{(i)}, t^{(i)})$
    Update θ by using gradient descent rule
    `// ` ∇_θ `coming from BP alg.`
    $\theta^k = \theta^{k-1} - \eta \nabla_\theta L(f(x_i, \theta), y_i))$
  return θ^k

In Algorithm 2, ∇_θ represents the partial derivative of the loss function according to the selected parameter of θ, which requires the partial derivatives of the nonlinear composite activation functions. In other terms, the training algorithm searches for the optimal values of the learnable parameters to minimise 2.19.

As an example objective function, we can define binary cross-entropy (BCE) loss function, which is defined as negative-log likelihood using each pair of (y_i, t_i) to maximise the correct estimation or minimise the BCE error between predicted y^i and target data t^i as follows:

$$L_{BCE} = -\frac{1}{N_{out}} \sum_{i=1}^{N_{out}} t_i \log(y_i) + (1-t_i) \log(1-y_i) \tag{2.19}$$

or mean squared error (MSE) $\mathrm{L}_\zeta{}^* = L_{MSE}$ can be defined as follows:

$$\begin{aligned} \mathrm{L}_\zeta{}^* &= \frac{1}{N} \sum_{i=1}^{N} \mathrm{L}_\zeta(y_i, t_i) \\ &= \frac{1}{N} \sum_{D} \frac{1}{2} \parallel t_i - y_i \parallel_2 \end{aligned} \tag{2.20}$$

$$\begin{aligned} \frac{\partial L_{MSE}}{\partial w_{ij}^{(k)}} &= \frac{\partial}{\partial w_{ij}^{(k)}} \left(\frac{1}{N} \sum_{\mathcal{D}} \left[\frac{1}{2} \parallel t_i - y_i \parallel_2 \right] \right), \\ &= \frac{1}{N} \sum_{\mathcal{D}} \left[\frac{1}{2} \frac{\partial}{\partial w_{ij}^{(k)}} \left[(t_i - y_i)^T (t_i - y_i) \right] \right], \\ &= \frac{1}{N} \sum_{\mathcal{D}} \left[(t_i - y_i)^T \frac{\partial (-y_i)}{\partial w_{ij}^{(k)}} \right]. \end{aligned} \tag{2.21}$$

Here $w_{ij}^{(k)}$ is a weight vector from neuron i to j neuron at layer k. Next section briefs the previous neural language models.

2.3 Neural Language Models

Language Models aim to learn likelihood of a each word in a given collection or context. Recently, neural network language models have became powerful tools with the combinations distributed representations of words, which is also called as word embeddings, word vectors or distributed feature vectors throughout the thesis interchangeably.

Neural Language Models (NLMs) are basically performed for learning high-quality representations of sentences from words. They are designed for computing word embeddings based on the syntactic and semantic similarities, which are basically feedforward networks designed with hidden layers. NLMs designed to learn probability of a word $P(w_t|c)$ using previous given context. The context words $(w_{t-3}, w_{t-2}, w_{t-1})$ are concatenated and used as an input to the second hidden layer. The output layer estimates the word w_t using

softmax. Figure 2.7 shows the selected 3-gram context words to predict the word w_t. For each word in database, the matrix E is used to represent context embedding. They are distributed representations of context words, which are calculated by embedding tools, that will be explained in the next section. The embedding vectors concatenated to one vector using projection layer as can be seen in Figure 2.7. The forward pass of NLMs algorithm is given at Algorithm 3.

A bottom-up fully-connected architecture of neural language model network in the Figure 2.7 is designed with input layer, 2 hidden layers (projection + hidden) and output layer to predict the word "izmir" according to previous 3-gram context words "şehirlerden, İstanbul, Ankara". The projection layer concatenates the embeddings of context words into $1 \; x \; 3d$, where d represents the embedding vector dimension coming from shared word embeddings matrix E. The rows of the matrix E represent the dense feature vectors of whole words in the vocabulary. Inputs are one-hot encoding vectors with dimension $1 \; x \; |V|$, which are converted dense real-valued distributed embedding vectors to give as an input to the projection layer. *Distributional hypothesis* has been a foundation approach

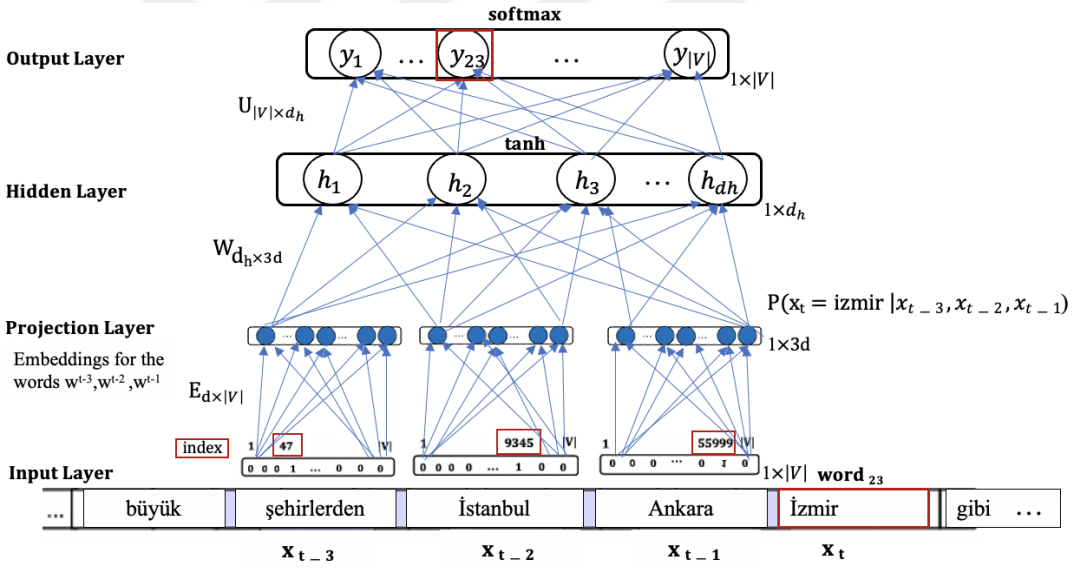

Figure 2.7 A bottom-up fully-connected neural language model

Algorithm 3: Feedforward neural network language model

Input: contexts= $\{w_{t-n+1}, \ldots, w_{t-1}\}$, vocabulary,
weight matrices= E, W, U , $bias = b$
$e \leftarrow [E(w_{t-n+1}), \ldots, E(w_{t-1})]$
$h \leftarrow tanh(We + b)$
$y \leftarrow softmax(Uh)$
return y

for many studies focused on learning word vectors using context. Class-based n-grams

models [25], Latent Semantic Analysis [26], a simple Recurrent Networks [27] are the previous key studies during 1990s. The neural language models have been adopted based on these foundation studies aiming to capture distributional semantics. Bengio *et al.* have been designed the neural probabilistic language model as a foundation study for learning continuous semantic representations of word vectors [28]. The authors proposed a simple Feedforward Neural Network (FFN) model dealing with curse of the dimensionality problem.

2.3.1 Distributed Representations

Before the deep learning era, the meaning of words conventionally learned using external lexical resources like Wordnet [11]. Word embeddings or distributed representations of words have become very popular recently as a feature learning technique that calculates the real number of continuous vectors of words in a low-dimensional space. They have constructed according to distributional hypothesis [29] and can demonstrate words semantically and syntactically using embedded continuous representations [30–33].

Distributional techniques in vector space models have a long history in theoretical linguistics [34], and they provide the basic construction of semantic vector embedding with the fact that similar words lie close in the vector space [35]. *Vector Space Models* [25] also called *Semantic Spaces* or *Distributional Models* aims to project the words from text corpus to an n-dimensional vector space by using the basis of *Distributional Hypothesis*, *"Words that occur in similar contexts tend to have similar meanings"* [36]. Based on this, words are represented as real-valued vectors in terms of context meanings and spatial proximity (see Figure 2.8). This approach is also the starting point of constructing semantic vector space models [37] satisfying identity, additivity, multiplicity, distributivity axioms [38], which makes it possible to do mathematical operations with words and map them into vector space according to their meanings.

Let V denotes the set of words that is called as a *dictionary*, *vocabulary* or *database*. Each word is defined by a real-valued *n-dimensional* numerical elements or objects of the vector space (semantic space). The function $d : V \times V \mapsto R$ is a distance or similarity measure between words. The meaning of words is represented as a part of high dimensional semantic space, and words can be compared by using standard similarity or distance measures. A *similarity measure* or *similarity function* is a function of two variables that quantified the similarity between two objects. For example, *cosine similarity measure* between n-dimensional vectors x and y is calculated by the cosine of the angle

Figure 2.8 Examples for semantic word embeddings

between them as follows: $x = \{x_1, x_2, \ldots, x_N\}$ and $y = \{y_1, y_2, \ldots, y_N\}$ is defined as:

$$\text{sim}(x,y) = \frac{x \cdot y}{|x||y|} = \frac{\sum_{i=1}^{N} x_i y_i}{\sqrt{\left(\sum_{i=1}^{N} x_i^2\right)\left(\sum_{i=1}^{N} y_i^2\right)}}$$

Here words are represented as an ordered list of real numbers $v = (v_1, v_2, \ldots, v_{n-1}, v_n)$ and v_i is the i^{th} component of the word vector.

Assume that $x \in \mathbb{R}^d$ is a d-dimensional input vector. The hidden layer output is evaluated as follows:

$$\begin{aligned} h_a &= (W_{ih}\, x + b_{ih}) \\ y &= \varphi(h_a) \\ S_y(x) &= {W_{hy}}^T\, y \end{aligned} \tag{2.22}$$

$\nabla(S_y(x))|_\theta$ is needed to be calculated to learn the optimum value of $\theta = W_{hy}$. The partial derivative of the output score is calculated by chain-rule as follows:

$$\begin{aligned} \frac{\partial S_y(x)}{\partial W_{hy}} &= \frac{\partial {W_{hy}}^T\, y}{\partial W_{hy}} \\ \frac{\partial S_y(x)}{\partial W_{hy}} &= y \end{aligned} \tag{2.23}$$

$$\begin{aligned}\frac{\partial S_y(x)}{\partial W_{ih}} &= \frac{\partial {W_{hy}}^T\, y}{\partial W_{ih}} \\ &= \frac{\partial W_{hy}\, \varphi(W_{ih}\, x + b_{ih})}{\partial W_{ih}}\end{aligned} \tag{2.24}$$

If ${W_{ih}}^{kj}$ is considered one single weight of k^{th} hidden layer of the j. hidden neuron, the full gradient of W_{ih} is calculated by chain-rule as follows:

$$\begin{aligned}\frac{\partial {W_{hy}}^T y}{\partial {W_{ih}}^{kj}} &= \frac{\partial {W_{hy}}^{kj}\, y^j}{\partial {W_{ih}}^{kj}} \\ &= {W_{hy}}^{kj} \frac{\partial y^j}{\partial {W_{ih}}^{kj}} \\ &= {W_{hy}}^{kj} \frac{\partial y^j}{\partial h^j} \frac{\partial h^j}{\partial {W_{ih}}^{kj}} \\ &= {W_{hy}}^{kj} \frac{\partial \varphi(h^j)}{\partial h^j} \frac{\partial h^j}{\partial {W_{ih}}^{kj}} \\ &= {W_{hy}}^{kj} \varphi'(h^j) \frac{\partial h^j}{\partial {W_{ih}}^{kj}} \\ &= {\delta_{hy}}^j \frac{\partial}{\partial {W_{ih}}^{kj}} \sum_i {W_{ih}}^{ji} x^i \\ &= {\delta_{hy}}^j x^i\end{aligned} \tag{2.25}$$

$$\begin{aligned}\frac{\partial S_y(x)}{\partial b_{ih}} &= {W_{hy}}^{kj} \varphi'(h^j) \frac{\partial h^j}{\partial b_{ih}} \\ &= {W_{hy}}^{kj} \varphi'(h^j) \frac{\partial W_{ih} x + b_{ih}}{\partial b_{ih}} \\ &= {\delta_{hy}}^j\end{aligned} \tag{2.26}$$

where ${\delta_{hy}}^j$ is the local error signal and x^i is the local input signal.

2.3.2 Static Word Embeddings

This section reviews the static (context-free) embedding models. Word embeddings have been used as a practical tool in neural network models and construction of Language Models (LMs) for many tasks including SA [4], QA [39, 40] and named entity recognition (NER) [41]. They enable the representation of word-level features from massive datasets into low-dimensional Euclidean space vectors.

Word vectors (word2vec) are a well-known window-based word vector learning models [28], [30], [33] to learn distributional representations. Skip-Gram (SG) model is one of

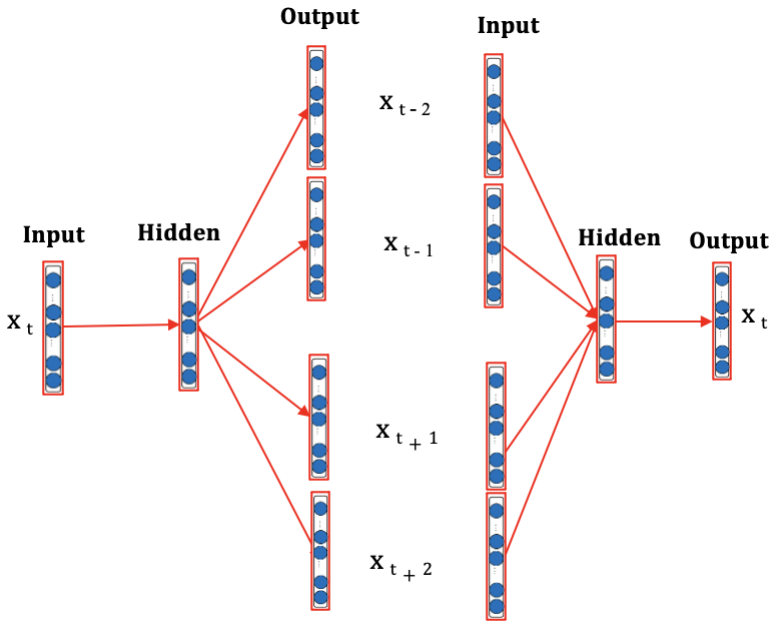

Figure 2.9 Architecture of the skip-gram and CBOW model with window size $c = 2$

the well-known window-based static neural word embedding algorithms. It aims to learn the representation of context (neighbours of the centre word) using a given target word. The neighbourhood of the target words are claimed as positive class, and the other words of the vocabulary are claimed as negative samples.

Let $x_w \in R^n$ be the n-dimensional word embedding of token w in a given vocabulary V. Given a pair of words $(w_t,\ w_c)$, the likelihood of the word x_c is probability of the observation of the word x_c using the target word x_t, which is defined as follows:

$$P(w_c|w_t) = \frac{\exp({x_c}^T x_t)}{\sum_{i=1}^{|V|} \exp({x_i}^T x_t)} \tag{2.27}$$

Here x_t represents target word, x_c represents the context words the target vector. The word embeddings of the vocabulary $x_1, x_2, \ldots, x_T$ are learned by maximizing the following objective function:

$$L_\zeta(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{-c \le k \le c} \log P(w_{k+t}|w_t) \tag{2.28}$$

The context-window contains equal word vectors from the left and right of the targeted word vector. For a targeted word vector x_t, context window defined as $x_c = \{x_{t-k}, \ldots, x_{t-1}, x_{t+1}, \ldots, x_{t+k}\}$ where $k \in \{1, 2, \ldots, N\}$.

Similar to SG, Continuous Bag-of-Words (CBOW) are used the produce word embedding as a numeric distributional representation of words. It aims to find a higher prediction of a specific word by using context as an input. Both of them are unsupervised learning models that are used a large input corpus for the training phase and produce word vectors for each of the words in the corpus. Figure 2.9 represents the architecture of the SG and CBOW model with window size $c = 2$.

There are some other learning models like doc2vec, which represents words and the whole sentence and documents. Doc2vec models have proposed by inspiration and extension of the word-level representations. The common way to calculate the whole sentence is by calculating the average word embeddings [42]. Mikolov and Le have also proposed paragraph vector method to find a dense vector of sentences, paragraphs, and documents [42]. This approach has experimented with the Stanford Sentiment Treebank (SST), and IMDB sentiment analysis datasets [42]. The experiments showed that the doc2vec model has the potential to overcome many word2vec models, especially many weaknesses of the bag-of-words models. However, this approach causes the loss of relevant information about the order of words and ignore the semantics of the words.

FastText has been implemented to handle rich word morphology. Especially the word vectors for MRLs like Turkish, the words with the same root such as *gözlükçü* "optician" and *gözlük* "glasses" don't have similar word vectors. Bojanowski *et al.* suggested calculating representation by combining the character-level features and SG model. The model can represent words as a sum of its character n-grams, so construction of compound words getting easy especially for MRLs. Additionally, this model provides fast and robust text classification by using the model probability of a label, and it has the pre-trained word vectors for scarce supervised data [43].

2.3.3 Recurrent (Chain-Structured) Language Models

Sentence embeddings can be calculated using word embeddings as an inputs to the NLMs. The traditional MLP networks cannot capture the sequential structure in the data like sequence of words observations. In this section, we focus on conventional chain-structured models, which have been used to learn from variable-length, temporal or sequential inputs. Firstly, the deep recurrent neural network (RNN) model has been described based on the standard (vanilla) RNN model for the SA task. Since we focus on the SA task, we try to optimise the CE or BCE loss function. Required optimisation steps based on the training algorithm are also given. The training issues of the many-to-one deep recurrent learning models have also been discussed in this section.

2.3.3.1 Deep Recurrent Neural Networks

Recurrent Neural Networks (RNNs) have implemented to construct the sentences, paragraphs and documents representations sequentially. Theoretically, they can transfer information to sequence steps between long time dependencies and even model unlimited length data. The simplest version of RNNs is called as Elman network [27], which has an input layer, a hidden layer, and an output layer.

The recurrent neural layers of a deep RNN are dynamical connectionist layers, that can learn from a sequential temporal data. Let $x^{(i)} = (x_1,\ x_2,\ \ldots, x_{t-1},\ x_t)$ be a sequential feature vector of the sequence of words, i.e. n-dimensional word embeddings or word vectors for each observation in the dataset. The deep RNN model has been constructed using the sequences of the following recurrence formula and defined below for each time step t with two commonly used nonlinear activation functions, tanh and sigmoid as follows:

$$h_t = tanh(W_{hh}\ h_{t-1} + W_{xh}\ x_t + b_h) \tag{2.29}$$

$$y = sigmoid(W_{hy}\ h_t + b_y) \tag{2.30}$$

Figure 2.10 represents a single RNN cell which is many-to-one RNN model and unfolding it for time step t. The hidden state of the model h_t passes the information from the previous time step h_{t-1}, and uses it to classify the given observation $x^{(i)}$. The sigmoid function is used to predict the sentiment class of $x^{(i)}$, i.e. each review or tweet from the dataset. The predicted observation is calculated by classifier function, $g : x^{(i)} \rightarrow y^{(i)} \in \{0, 1\}$, one for positive sentiment class and zero for negative sentiment class using a supervised learning algorithm. The learning happens by updating $W_{xh},\ W_{hh},\ W_{hy}$ which are shared learnable weight parameters. Besides, b_h and b_y are the learnable bias terms that are used for the interception. The training algorithms, such as SGD, have been used to learn the optimal values of system parameters to minimise the loss function or the objective cost function. The proposed method used the binary cross-entropy (BCE) loss function, which is defined as negative-log likelihood using each pair of $(y^{(i)},\ t^{(i)})$ to maximise the correct estimation or minimise the BCE error between predicted $y^{(i)}$ and target data $t^{(i)}$ as defined in Equation 2.31:

$$L_{BCE} = -\frac{1}{N_{out}} \sum_{i=1}^{N_{out}} t_i \log(y_i) + (1 - t_i) \log(1 - y_i) \tag{2.31}$$

Algorithm 4 is the pseudo-code of the forward pass for one sample $(x^{(i)},\ t^{(i)})$ pair from the dataset. Functions f and g represent the nonlinear tanh and sigmoid activation functions, and the total loss function calculated from the summation of each pair loss as defined in Equation 2.31. The stochastic training algorithms learn from one example pair

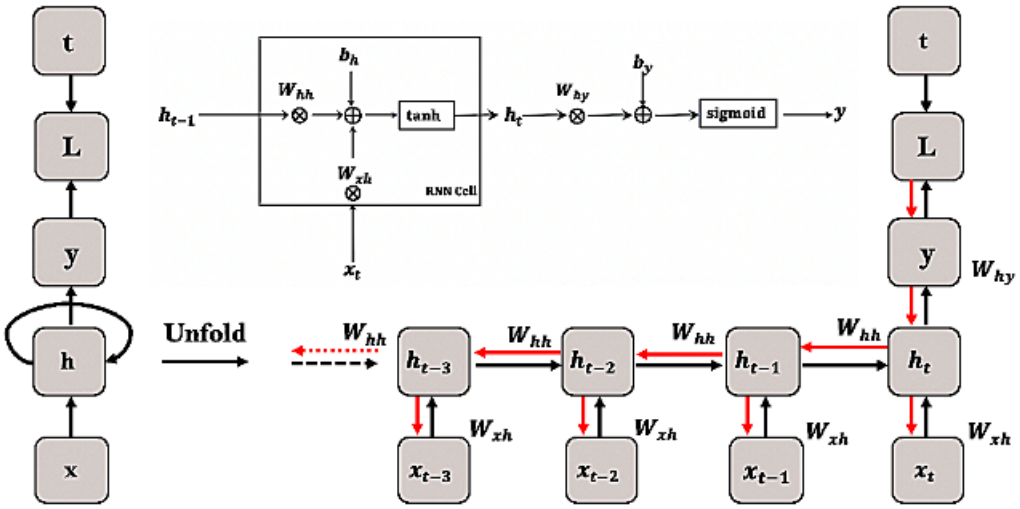

Figure 2.10 A single RNN cell which is many-to-one RNN model (left), and unfolding it for time step t (right)

Algorithm 4: Forward propagation of the RNN

Input: temporal sequential input $x^{(i)} = (x_1, x_2, \ldots, x_{t-1}, x_t)$,
learnable parameters: $\theta = (W_{xh}, W_{hh}, W_{hy}, b_h, b_y)$
desired output target: t
Output: Predicted output: y and Loss : L
Function `ForwardPass` (x, t, θ) **:**

Initialize hidden states
$h_0 \leftarrow 0$
for $i \leftarrow 1$ ***to*** T **do**
  $h_t \leftarrow f(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$
$y \leftarrow g(W_{hy}h_t + b_y)$
$L \leftarrow L(y,t)$
return y, L

at each iteration instead of using the entire dataset at each step, which is a particularly useful method for the large training datasets as defined in Algorithm 2. The gradient-based training algorithms use backpropagation algorithm, which is calculating the partial derivatives according to learnable parameters of the system via the chain rule. For deep RNN backpropagation through time (BPTT) is applied as in Algorithm 5.

The BPTT is applied by the chain rule to compute the partial derivatives of the learnable parameters of θ through time as given in Algorithm 5. At each training time step of the conventional gradient-based algorithms, the same shared weights are used during backpropagation; hence the gradients can easily vanish or explode as discussed in section 2.4.1. The gradient descent and backpropagation through time algorithms are combined to train deep RNN models. As a powerful generalisation of the gradient descent strategy, the stochastic gradient descent is commonly used to train large dataset. In the Algorithm

Algorithm 5: Backpropagation through time (BPTT)

Input: Loss: L, $x^{(i)} = (x_1, x_2, \ldots, x_{t-1}, x_t)$,
$\theta = (W_{xh}, W_{hh}, W_{hy}, b_h, b_y)$
desired output target: t
Output: $d\theta = (dW_{xh}, dW_{hh}, dW_{hy}, db_h, db_y)$
Function `BPTT(`L, θ`)`**:**

- **for** $t \leftarrow T$ ***to*** 1 **do**
  - $dW_{hy} \leftarrow \frac{\partial L}{\partial y}\frac{\partial y}{\partial W_{hy}}$
  - $db_y \leftarrow \frac{\partial L}{\partial y}\frac{\partial y}{\partial b_y}$
  - $dW_{hh} \leftarrow \frac{\partial L}{\partial y}\frac{\partial y}{\partial h_t}\frac{\partial h_t}{\partial W_{hh}}$
  - $dW_{xh} \leftarrow \frac{\partial L}{\partial y}\frac{\partial y}{\partial h_t}\frac{\partial h_t}{\partial W_{xh}}$
  - $dW_{hy} \leftarrow \frac{\partial L}{\partial y}\frac{\partial y}{\partial W_{hy}}$
- **return** $d\theta = (dW_{xh}, dW_{hh}, dW_{hy}, db_h, db_y)$

6, the steps are given for SGD algorithm.

Algorithm 6: BPTT combined with SGD

Input: Learning rate:η, $\theta = (W_{xh}, W_{hh}, W_{hy}, b_h, b_y)$
Output: θ^{t+1}
Function `SGD(`η, θ`)`**:**

- **for** *each training step t* **do**
  - Select training pairs $(x^{(i)}, t^{(i)})$
  - Update θ by using gradient descent rule
  - `// `∇_θ` coming from BPTT alg.`
  - $\theta^{t+1} = \theta^t - \eta \nabla_\theta L(f(x_i, \theta), y_i))$
- **return** θ^{t+1}

In Algorithm 6, ∇_θ represents the derivative of the loss according to the selected variable of θ, which requires the partial derivatives of the nonlinear composite activation functions. In other terms, the training algorithm searches for the optimum rates of the θ to minimise 2.31.

Here η is the learning rate, that is one of the most important hyperparameters of the deep learning models. Once the loss function is calculated at the feedforward step, the proposed partial derivatives $\frac{\partial L}{\partial W_{hy}}$, $\frac{\partial L}{\partial b_y}$, $\frac{\partial L}{\partial W_{hh}}$, $\frac{\partial L}{\partial W_{xh}}$, $\frac{\partial L}{\partial b_x}$ represented as ∇_θ in the Equation 2.33 given in section 2.4.1 need to be calculated for updating the set of trainable parameters of the system. During the backpropagation, since the same weight parameters are shared at each time step, the recursive nature of the training process causes the *vanishing and exploding gradients (VEG) problem* which causes a huge loss of knowledge across the deep hidden layers that we will discuss it section 2.4.

2.3.3.2 Long Short-Term Memory (LSTM) Networks

LSTM is a specific version of the recurrent language model which was designed to model the long-term dependencies within random-sized sequential data [44], [45]. The information across the chain (i.e. sequence) structured LSTM layers be transferred to learn using following equations:

$$\begin{aligned} ig_t &= \varphi\left(W_{ih}\, x_t + U_{ih}\, h_{t-1} + b^i\right) \\ fg_t &= \varphi\left(W_{hf}\, x_t + U_{hf}\, h_{t-1} + b^f\right) \\ og_t &= \varphi\left(W_{ho}\, x_t + U_{ho}\, h_{t-1} + b^o\right) \\ ug_t &= \psi\left(W_u\, x_t + U_u\, h_{t-1} + b^u\right) \\ mc_t &= ig_t \odot ug_t + fg_t \odot mc_{t-1} \\ hg_t &= og_t \odot \psi(mc_t) \end{aligned} \tag{2.32}$$

Here the entries of ig_t, fg_t, $og_t \in [0,1]$. Learning happens by processing inputs at every time step in a sequential composition operations of the sigmoid function, which is represented by φ. Forget layer is the gate to decide whether the information coming from the memory cell should be forgotten. Unifying layer combines the information using tanh function represented by ψ. Learning happens by updating the learnable parameters $W_{ih}, U_{ih}, W_{hf}, U_{hf}, W_{ho}, U_{ho}, W_u, U_u, b^i, b^f, b^o, b^u$. Figure 2.11 represents the basic chain structured LSTM cell.

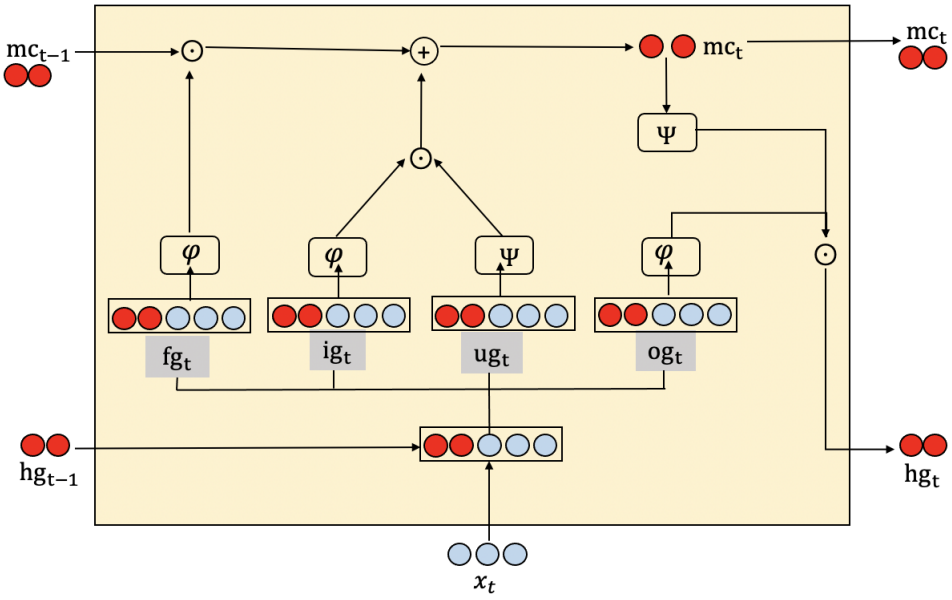

Figure 2.11 Representation of the basic chain-structured LSTM cell

2.4 Training Difficulties of Deep Recurrent Models

2.4.1 Vanishing and Exploding Gradients (VEG) Problem

Most of the training algorithms, such as SGD are based on the gradient descent learning rule by backpropagation through time using following update rule for each time step t:

$$\theta^{t+1} = \theta^t - \eta \ \nabla_\theta \ L(f(x_i, \theta), t_i)) \tag{2.33}$$

Here η is the learning rate, which is a critical hyperparameters of the learning system. Once the loss function is calculated at the feedforward step, the proposed recurrent neural network model is required to calculate the partial derivatives $\frac{\partial L}{\partial W_{hy}}$, $\frac{\partial L}{\partial b_y}$, $\frac{\partial L}{\partial W_{hh}}$, $\frac{\partial L}{\partial W_{xh}}$, $\frac{\partial L}{\partial b_x}$ that represented as ∇_θ in the Equation 2.33 above for updating the set of trainable parameters of the system. The recursive nature of these process causes "*the vanishing and exploding gradients (VEG) problem*", which is formulated in detail below:

$$\begin{aligned}
L(t,y) &= \sum_{i=1}^{N} L_t(y_i, t_i) \\
\frac{\partial L}{\partial W_{hh}} &= \sum_{t=1}^{N_{out}} \frac{\partial L_t}{\partial W_{hh}} \\
&= \sum_{t=1}^{N_{out}} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial W_{hh}} \\
\frac{\partial h_t}{\partial W_{hh}} &= \ tanh'_t(W_{hh} h_{t-1} + W_{xh} x_t + b_h)[h_{t-1} + \ W_{hh} \frac{\partial h_{t-1}}{\partial W_{hh}}]
\end{aligned} \tag{2.34}$$

h_t and h_{t-1} are both a function of W_{hh}, so the product rule of the derivative should be used for every time step as follows:

$$\frac{\partial h_{t-1}}{\partial W_{hh}} = tanh'_{t-1}(W_{hh} h_{t-2} + W_{xh} x_{t-1})[h_{t-2} + \ W_{hh} \frac{\partial h_{t-2}}{\partial W_{hh}}] \tag{2.35}$$

$$\frac{\partial h_{t-2}}{\partial W_{hh}} = tanh'_{t-2}(W_{hh} h_{t-3} + W_{xh} x_{t-2})[h_{t-3} + \ W_{hh} \frac{\partial h_{t-3}}{\partial W_{hh}}] \tag{2.36}$$

The rightmost term of the Equation 2.36 should be expanded until t = 1 to calculate $\frac{\partial h_1}{\partial W_{hh}}$, and the following backpropagated sequence is found if $tanh'_t(W_{hh}h_{t-2} + W_{xh}x_{t-1} + b_h)$ is represented as $tanh'_t$:

$$\begin{aligned}\frac{\partial h_t}{\partial W_{hh}} &= tanh'_t \left[h_{t-1} + W_{hh} tanh'_{t-1} \left[h_{t-2} + \ldots + W_{hh} \frac{\partial h_1}{\partial W_{hh}} \right] \right] \\ &= tanh'_t \; h_{t-1} + tanh'_t \; W_{hh} tanh'_{t-1} h_{t-2} + \ \ldots\end{aligned} \tag{2.37}$$

Here the partial derivatives of h_t are calculated with respect to the previous time step h_k. Therefore, the total loss can backpropagated according to W_{hh} as follows:

$$\frac{\partial L}{\partial W_{hh}} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_{hh}} \tag{2.38}$$

Here $\frac{\partial h_t}{\partial h_k} = \prod_{s=k+1}^{t} \frac{\partial h_s}{\partial h_{s-1}}$ for any time state s of the system, and each $\frac{\partial h_s}{\partial h_{s-1}}$ is the Jacobian matrix for h $\in \mathbb{R}^{D_n}$ defined as follows:

$$\begin{aligned}\frac{\partial h_s}{\partial h_{s-1}} &= \begin{bmatrix} \frac{\partial h_s}{\partial h_{s-1,1}} \cdots \frac{\partial h_s}{\partial h_{s-1,D_n}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial h_{s,1}}{\partial h_{s-1,1}} & \cdots & \frac{\partial h_{s,1}}{\partial h_{s-1,D_n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_{s,D_n}}{\partial h_{s-1,1}} & \cdots & \frac{\partial h_{s,D_n}}{\partial h_{s-1,D_n}} \end{bmatrix}\end{aligned} \tag{2.39}$$

Equation 2.38 becomes the following:

$$\frac{\partial L}{\partial W_{hh}} = \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \Big(\prod_{s=k+1}^{t} \frac{\partial h_s}{\partial h_{s-1}} \Big) \frac{\partial h_k}{\partial W_{hh}} \tag{2.40}$$

Since $h_t = \ tanh \ (W_{hh}h_{t-1} + W_{xh}x_t + b_h)$, the term $\prod_{s=k+1}^{t} \frac{\partial h_s}{\partial h_{s-1}}$ gives the diagonal matrix from 2.39 and 2.40, that can be seen from the following equation :

$$\prod_{s=k+1}^{t} \frac{\partial h_s}{\partial h_{s-1}} = \prod_{s=k+1}^{t} W_{hh}^T diag(tanh'\left(W_{hh}h_{s-1}\right)\ ) \tag{2.41}$$

Let a and b be both the upper bounds, or largest singular values of the matrices W_{hh}^T and $diag(tanh'\left(W_{hh}h_{s-1}\right))$, respectively. The 2-norms of these matrices are bounded by ab defined as follows for any time state s of the system:

$$\left\| \frac{\partial h_s}{\partial h_{s-1}} \right\| = \left\| W_{hh}^T \right\| \left\| diag\left(tanh'\left(W_{hh}h_{s-1}\right)\right) \right\| \leq ab \tag{2.42}$$

$$\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{s=k+1}^{t} \frac{\partial h_s}{\partial h_{s-1}} \right\| \leq (ab)^{t-k} \tag{2.43}$$

During training, the same weight matrix W_{hh} is used across the layers, so as $t \to \infty$ the term $(a.b)^{t-k}$ vanishes at the very small value such as $(ab)^{t-k} \to 0$ or explodes to the extremely large value such as $(a.b)^{t-k} \to \infty$.

It has been shown that [46, 47], if the absolute value of the largest eigenvalue of the W_{hh}^T is smaller than $\frac{1}{b}$, then the gradients vanish as follows:

$$\begin{aligned}
\forall\ s, \left\| \frac{\partial h_s}{\partial h_{s-1}} \right\| &\leq \left\| W_{hh}^T \right\| \left\| diag\left(tanh'\left(W_{hh}h_{s-1}\right)\right) \right\| \\
&\leq ab < \frac{1}{b}.b < 1 \\
\exists\ \gamma, \left\| \frac{\partial h_s}{\partial h_{s-1}} \right\| &\leq \gamma < 1 \\
\left\| \frac{\partial h_t}{\partial h_k} \right\| &= \prod_{s=k+1}^{t} \frac{\partial h_s}{\partial h_{s-1}} \leq (\gamma)^{t-k}
\end{aligned} \tag{2.44}$$

As $t \to \infty$, it is clear that $\lim_{t\to\infty} \prod_{s=k+1}^{t} \frac{\partial h_s}{\partial h_{s-1}} = 0$. Similarly, when the largest eigenvalue of the W_{hh}^T is bigger than $1/b$, gradients *explode* and $\lim_{t\to\infty} \prod_{s=k+1}^{t} \frac{\partial h_s}{\partial h_{s-1}} = \infty$. From the perspective of the dynamical systems, when the model keeps its state in the same stable state for a long time, the information about the system updating is lost [47]. Hence, more practical and efficient approaches are needed to train deep RNN architectures which are mostly modelled with the nonlinear activation functions.

2.4.2 Approaches to Handle VEG Problem

This section reviews the approaches that have been used to handle the VEG problem in training deep RNN models.

2.4.2.1 An Alternative Deep Architectures

The first approach to handle the VEG problem is to use newer types of RNN architectures such as LSTMs [44], Gated-Recurrent Units (GRUs) [48] and Echo-State-Networks (ESNs) [49]. These architectures can model sequences and they produced good results for many applications [50]. But, they have issues such as limited non-linearity learning abilities [51], training times that can sometimes be many days or even months and are still not completely free from the same gradient problem. Besides, hyperparameter optimisation and initial parameter tuning are also needed to improve their performances. As a similar way to the using of new architectures, some researchers have proposed to use new activation functions such as ReLU instead of hyperbolic tangent or sigmoid functions [52–54]. However, these approaches have limitations as well. For example, as the ReLU function is positive definite, it causes a bias shift effect, which means that it behaves like a bias term for the next layers and decreases generalization capacity [55].

2.4.2.2 Gradient Stabilisation Methods

The second way to handle the vanishing and exploding gradient problem is the stabilisation of the updated recurrent weights. Gradient clipping [46] is a well-known heuristic approach to rescale gradients. It controls the direction in parameter updating by using a given threshold and prevents unexpected falls to zero or rises to infinity before operating the gradient-descent learning rule. L_1 and L_2 regularisation has also been applied to the recurrent weights to prevent overfitting. They are used as a penalty term during training mainly to bring weights closer to zero [56]. Initialisation methods have also been employed to limit the values of the updated parameters by using the identity or orthogonal shared matrix [57–59]. SVD has been used to find the orthogonal matrices of the weight matrix, and they proposed to update the parameters at each iteration by using geodesic GD and Cayley methods [59]. However, these methods require the computation of inverse matrices, which sometimes can be complex. Additionally, the unitary initial matrices cannot be held after many training iterations, and the same issues arise again.

2.4.2.3 Hessian-free and Gradient-free Methods

In addition to the aforementioned approaches, Hessian-free (HF) optimisation methods [51] have been proposed to model the curvature of the nonlinear functions of deep RNN

models using random initialisation. They have been inspired by the second-order derivative method and Newton optimisation method, which are also called as a truncated-Newton or the pseudo-Newton method [60]. They have been applied successfully to large scale architectures [61], but as [62] mentioned that, mentioned that besides their sophisticated nature they have not enough generalisation ability to learn and need additional damping among hidden layers [61].

Some gradient independent methods have also been developed to tackle training difficulties of the RNN. One of the first most well-known heuristic search approaches is the Simulated Annealing method, which performs a random neighbourhood search iteratively to find optimal weights of the system [47]. However, as mentioned in [47], since its training time could be too long, alternative practical training methods are still needed.

2.4.2.4 Metaheuristics

So far, metaheuristic algorithms have been successfully implemented to find optimal solutions for complicated nonlinear optimisation problems. They have good initialisation strategies and local search abilities that bring crucial advantages to handle local optima issues such as getting trapped at local optima [63]. Although the number of studies for optimisation of deep architectures is less than that for conventional architectures [64–68], some studies have been carried out to improve the optimisation performances for the specified tasks using the intelligent nature of population-based algorithms. Those studies are mainly focused on hybridisation approaches which are used to evolve deep architectures and to optimise the hyperparameters of the deep learning models. Ge *et al.* have presented the modified Particle Swarm Optimisation (MPSO) [69] algorithm for training dynamic recurrent Elman networks [70], which try to learn the network structure and its parameters initially for controlling Ultrasonic Motors. Xiao *et al.* [71] have proposed a hybrid training algorithm with PSO and backpropagation (BP) for Impedance Identification. The RNN architecture has been trained based on finding the minimum MSE and the largest gradient. Zhang *et al.* have also proposed hybrid PSO and Evolutionary Algorithm (EA) to train RNN for solar radiation prediction [72]. Likewise, Cai *et al.* [73] have used hybrid PSO-EA for time series prediction with RNN. A continuous GA implemented for training RNN by updating weight parameters using random real-valued chromosomes [74].

Although the proposed hybrid approaches can train RNNs, those networks do not have so many hidden layers that they can be considered as deep architectures. Desell *et al.* have proposed Ant Colony Optimisation (ACO) [75] to evolve deep RNNs [64] and improve the architecture of the networks with artificial ants. Similarly, ElSaid *et al.* have also proposed employing the ACO algorithm to evolve the LSTM network structure [76]. Again, the networks developed do not comprise many hidden layers and cannot be regarded as

deep. More studies should be performed to explore the advantages of metaheuristic approaches for deep learning and to improve the deep recurrent learning abilities. Section 5 proposed a novel method to handle VEG problem, that occurs during training for SA task. Next section defines the SA task, levels and resources of SA, and discusses the issues of the SA in detail.

2.5 Levels, Resources and Issues of Sentiment Analysis

Sentiment analysis (SA) is based on detecting the polarity of a given text, which is generally classified as a positive, objective, or negative thinking about specific domains, subjects or items [77]. Before giving a formal definition let us consider the following review for a sentiment analysis:

> ***Review A:*** *User Sultan's Friend* ***Date:*** *April 9, 2021*
> *"(1) I am such a fan of the TV show F.R.I.E.N.D.S. (2) The characters are well developed independently and together they play off each other well. (3) Some moments are cheesy while others are funny, while the rest is just very well. (4) The actors and actresses portray their characters terribly fun. (5) The stories are unique and the script is amazing! (6) People are saying that friends is running out of ideas and that the humor is getting dull. (7) Running out of ideas? possibly. (8) Is humor getting feeble? absolutely not. (9) In fact, it's getting better and really captured my heart. (10) I know that they'll be there for me!!!"*

This review is about a movie, which it is called as a *target (g)* or *entity (g)* of an opinion. A *sentiment (s)* is a subjective expression about a target or entity. It can be *positive, negative, neutral or a rating score*. The user X is an *opinion holder (h)*, who post opinions about a related target at *time* t. Sentiment analysis mainly focus on the polar opinions. Hence more formally and broadly an *opinion* is defined as a quadruple (g,s,h,t).

Sentiment Analysis is an interdisciplinary research topic, which is related to linguistics, psychology, and computer science; which is also state as an *opinion mining* in academia [1]. Sentiments of an opinion can be *emotional* or *rational*. For example, the sentences *"This book is worth to read"* and *"The battery life of this computer is not long"* contains a rational sentiments about the book and battery life of computer targets respectively. The sentences also can express emotional feeling, such as *I am so happy with my excellent new car.*, *"Best restaurant and best foods ever"* or *What's wrong with you, are you kidding me?*. The terms *"affect, emotion and mood"* are also related psychological, physiological and sociological terms, which are interrelated terms. Affective computing is interested in detection of sentiments by face gestures generally and emotion analysis focus on the detection of the "anger, fear, joy, love, sadness, surprise" [78]. SA interested in processing natural language to explore how emotions and affects of emotions are expressed in text.

There are different levels of sentiment analysis in literature. If the overall polarity of the review which is expressed based on a single target is considered, this is the *document (review) level* SA, as TV show mentioned in review A. When detecting the sentiment is done only considering one sentence, it is the *sentence-level* SA. More precisely, *aspect-level* SA studies focus on a specific targeted feature of the product. For example, it could be rooms of the hotels, the service quality of the hotel or the food of the hotel for an hotel domain SA application. *Word-level* SA applications is looking for the sentiment orientation of each token (word) of the sentence, which is particularly usable for domain-specific applications.

The accuracy level for the sentiment analysis tasks has mainly related to the availability of sentiment analysis resources and NLP tools. For example, understanding of the polarity of the domain-specific reviews requires using domain-specific [79] polarity lexicons, enhanced feature extraction rules and annotation strategies. Most SA methods and NLP resources such as *polarity lexicons* [14, 15, 80], and sentiment treebanks [2], parsers have been first implemented for English [81]. Many resources such as dictionaries and corpuses are developed to be used in semantics and sentiment analysis studies. The corpus is the whole of the texts that enable researchers to collect written and oral texts as recorded in the electronic database and to describe different aspects of the language by researchers to make specific or general-purpose investigations. Wordnet is a database created by the English "Cognitive Science Laboratory of Princeton University" and is still being updated [11]. Turkish version of Wordnet is also under development [12]. SentiWordNet, on the other hand, assigns numerical emotion scores with a maximum of 1 as positive (Pos), negative (Neg) and neutral (Obj) to each word working in connection with the WordNet dictionary for emotion classification [82]. Since such databases contain a limited number of words, they need constant updating to decrease the calculation performance of the models used. However, since Turkish is an additive language, many different words can be derived by adding various suffixes to each word root, and as a result, the number of different words increases, the space size becomes very large, and the processing time takes too much time. In addition, since the data to be used during emotion analysis is raw data, many noise and spelling errors in the raw data are corrected during the preliminary preparation stage on the data set. However, this may cause some expressions that express emotion to be lost, especially in data sets containing user comments. These resources are generally used in a *lexicon-based* classifiers frameworks. They are mainly focused on using annotated corpora and manual feature engineering methods. Besides humans can have additional bias while annotating words, and they can add additional subjectivity to the classification, which makes the process more complicated.

There are some challenging points of SA. Let us consider the review A from user Sultan's Friend, which is about a sitcom comedy film. Even the general sentiment orientation of

the review A is positive, the sentences of reviews could have different polarity levels. Some of the sentences expressed the sentiment *implicitly* or *explicitly*. For example, the sentences *"(2) The characters are well developed independently and together they play off each other well. (5) The stories are unique and the script is amazing!"* contains the *explicit (direct)* aspect expressions using *well, unique and amazing* about the characters of film, their acting, scripts, and scenario. If *"(7) Running out of ideas? possibly. (8) Is humor getting feeble? absolutely not. (9) In fact, it's getting better and really captured my heart."* is considered the expressions about the scenarios of the film and humor expressed indirectly and in an *implicit (indirect)* way as in (10) *"I know that they'll be there for me!!!"*. The easiest way of detecting the polarity of a sentence is using detecting polar words such as "good, wonderful, and amazing, like " as positive sentiment words, or "bad, awful, dislike" for negative ones. Polar words (generally adjectives and adverbs) with *intensifiers* are generally used such as *"very, so, extremely, dreadfully, really, awfully, terribly"* to express the intensity. The point to be careful at this point is the compositional phrase *terribly fun* of the sentence *(4) The actors and actresses portray their characters terribly fun.* express a strong positive sentiment intensity while the "terribly" sometimes can be used with the same meaning as "badly". Likewise intensifiers, diminishers including *"slightly, pretty, a little bit, a bit, somewhat, barely"* are changed the intensity such as expressed in the sentence *"(7) Running out of ideas? possibly."*. Intensifiers and diminishers are important particularly for fine-grained (multi-class) SA applications.

Another issue to be aware of the review is the numbers of opinion holders. There could be more than one opinion holder in a review, which cause a confusion about the sentiment orientation about the review. The sentence *"(6) People are saying that friends is running out of ideas and that the humor is getting dull."* express the other peoples opinion and it is negative contrast to reviewer. In similar cases, the overall sentiment of the review is needed to be find. Additionally, the sentences using the *conjunctions* like "but, while, however, except that" etc. combines the opposite polar expressions such as given in the *"(3) Some moments are cheesy while others are funny, while the rest is just very well."*. These expressions are much more complicated and they require *compositional sentiment analysis*. There are also some *domain-dependent* issues, which occur for different applications of SA. The "analysis" part of the sentiment analysis interrelated with different levels of applications domains, including product reviews, social media comments like tweets, news and political reviews, or hotel and restaurant comments. Unfortunately, there is no one complete module for each language that can analyze all type of the dataset correctly. For example, the same polar words could contain the opposite polarity for different domains. The adjective "big" could mean positive as in *"The hotel rooms are big, comfortable, and very clean."* in the hotel domain, but it contains negativity when it is used in the product domain such as *"Its battery is very big and heavy."*. Hence

different strategies for each domain are needed to be implemented. Similarly, when the user-generated dataset is considered, i.e. for example, when noisy dataset from social media is used, domain-dependent issues occur. For example, additional preprocessing steps should be done to process tweets. Many challenges appear in the processing steps because of the user-generated data with many grammatical mistakes. Moreover, since they are short comments and generally express implicit sentiments with figurative expressions it is hard to detect the exact opinions from them. The retweeted tweets have not to be collected while constructing or to be cleaned since they may be not express the opinions, and many additional preprocessing steps have to be performed to clean collected tweets from hashtags, URLs, or mentions (i.e. @unibirmingham). There is also *domain-independent* issues of SA that can occur regardless of the application domain. Rhetorical expressions, sarcastic and ironic phrases are also complicated sub-field of SA. There are syntactic (linguistic) and semantic issues, that are encountered in almost any language. Even the study considers only binary polarity detection, it is hard to understand compositional meanings, rhetorical questions, negations, or ironic phrases regardless of the language [79]. They could happen in the text infrequently and detection of them is too hard since even understanding the figurative language is hard in spoken language due to the nature of the ironic and sarcastic sentences. They can contain contextual presumptions, and they require background knowledge of the topic. It is hard to make an exact distinction between the sentimental and ironic expressions which make SA more complicated [77]. In this thesis, we focus on detecting compositional binary and fine-grained sentiment classes for Turkish as a well-known morphologically rich language (MRL). Since we focus on Turkish language, there is also *language-dependent* linguistic issues, which have been discussed in section 2.6.

2.6 Turkish and Its Challenging Semantic Structure for Sentiment Analysis

This section addresses the challenging linguistic issues of Turkish for SA task particularly. According to the Ethnologue [83], Turkish is the most 20th spoken language in the world, as the most widely spoken Turkic language with over 85 million speakers [84]. The world atlas of language database [85] is defined Turkish as an outlier among the Eurasia languages, since it has high synthesis level i.e. up to eight or nine categories per word on average [86]. This means one word in Turkish can take over 30 inflections [87] and it can produce many derivational forms of a same root as given in Table 2.1.

Turkish is a morphologically rich language (MRL), which means it has a very productive word structure which are constructed like “beads-on-a-string”. Even one root word can have massive amount of derivational morphemes. For example, the longest word in

Table 2.1 Examples of Turkish verb “oku” with derivational and inflectional suffixes construct a different sentences in English

Root: oku Word Formations in Turkish	**Root: read** Word Formations in English
okuyorum	I am reading
okuyorsun	You are reading
okuyor	He/ she/ it reading
okuyoruz	We are reading
okuyorsunuz	You are reading
okuyorlar	They are reading
okuduk	We read
okudukça	As long as (somebody) reads
okumalıyız	We must read
okumadan	Without reading
okuman	Your reading
okurken	While (somebody) is reading
okuyunca	When (somebody) comes
okutmak	To cause (somebody) to cause (another person) to read

Turkish is *“Muvaffakiyetsizleştiricileştiriveremeyebileceklerimizdenmişsinizcesine”* consists 70 letters. It also has an free constituent order, hence general English Subject-Object-Verb sentence structure sequence not followed in Turkish. Hence same words with different orders can cause various semantics like given in Table 2.2.

Table 2.2 Turkish has an free constituent, hence same words with different orders can cause various different meanings

Ali bilgisayarı aldı.	Ali took computer.
Bilgisayarı Ali aldı.	It was Ali, who took computer.
Aldı Ali bilgisayarı.	Ali took computer (but was not really supposed to took it).
Aldı bilgisayarı Ali.	Ali took computer (and I was expecting that).
Ali aldı bilgisayarı.	Ali took computer (but someone else could also have took it.)
Bilgisayarı aldı Ali.	Ali took computer (but he could have took something else.)

In addition to the freedom of constituent structure and lackness of data sources, Turkish has many other linguistic issues for the NLP tasks. Since Turkish has multiple derivations for one word, many words might be constructed by adding various suffixes to the same word. This causes *"out-of-vocabulary (OOV)"* problem. OOV words that might be seen in a given dataset while searching over a vocabulary consisting of pre-defined limited word set. Even as [88] mentioned that over 1M word could be derived and from

only one word using additional suffixes in MRLs. Thus, the likelihood of many OOV words increases exponentially, and some words only rarely appear in the training dataset. Even word vectors can model the natural language by using distributional information, they cannot model the unknown words which are not in the dataset. Although they can capture the same syntactic and semantic relations like singular/plural word information like $x_{pencil} \sim x_{pencils}$, rare and unseen words cannot represent efficiently [89]. A typical elimination solution for the OOV words causes the loss of many information since even one Turkish might corresponds to almost a sentence in English. For example the Turkish word “bitirebileceksek” is translated as "if we will be able to finish (it)" in English, which is as long as like a sentence. [79] pointed out the additive structure of the Turkish words and mentioned the challenging points of Turkish in SA. The morphological analysis of Turkish word is important since it may provide a sentiment information of a word. For example, the token *sabır* "patience" in root form is converted to positive with particular derivational suffix -lı and converted to negative with particular derivational suffix -sız *sabır-sız* "impatient". Additionally, one word can has many possible morphological features as given in Table 2.3. Moreover, as it can be seen in Figure 2.12 word order and morphological structure of the Turkish and English is different. Hence even most of the NLP tools have been implemented for English, new algorithms are needed to implement for Turkish language due to the linguistic structure dependency.

Figure 2.12 Word order and morphological structure differences between English and Turkish

In addition to Turkish linguistics difficulties, sentiment analysis has syntactic (linguistic) and semantic issues, that are encountered in almost any language. Even if SA focuses on binary level polarity; it is hard to understand compositional meanings, rhetorical questions, negations, or ironic phrases regardless of the language [79]. Hence converting words to the one-hot encoding to the continuous dense encoding is not enough to un-

Table 2.3 The possible morphological analysis for the one word “arın” in Turkish

Roots / Category	Morphological Analysis
[arınmak:Verb]	arın:Verb+Imp+A2sg
[ar:Noun,Prop]	ar:Noun+A3sg+ın:Gen
[ar:Noun,Prop]	ar:Noun+A3sg+ın:P2sg
[ar:Noun]	ar:Noun+A3sg+ın:Gen
[arı:Noun,Prop]	arı:Noun+A3sg+n:P2sg
[arı:Noun]	arı:Noun+A3sg+n:P2sg
[arın:Noun,Prop]	arın:Noun+A3sg
[arı:Adj]	arı:Adj|Zero+Noun+A3sg+n:P2sg

derstand the continuous combinations of semantics in Turkish language. Additionally, when the user-generated dataset is considered, i.e., when noisy dataset from social media is used, domain-dependent issues occur. Hence, it is needed to be done time-consuming specialised preprocessing steps for cleaning a dataset. Moreover, the understanding of the polarity of the domain-specific reviews requires using domain-specific [79] polarity lexicons, enhanced feature extraction rules and annotation strategies. These are generally done by time-consuming manual processes, also humans can have additional bias while annotating words, and they can add additional subjectivity to the classification.

This thesis mainly focused on capturing the sentiment of longer phrases by combining morph level, stem level, word-phrase level and sentence level and review level sentiment information, with the implementations of various datasets from different domains, including movie reviews and various product reviews. Most previous studies have been done based on a binary prediction of a given sentence due to limited resources of a fine-grained labelled large dataset. Hence, a novel fine-grained sentiment classification dataset has been introduced to contribute to the lack of Turkish NLP resources. Section 3 presents the first proposed recursive model and the datasets in detail.

3
MS-TR: A MORPHOLOGICALLY ENRICHED SENTIMENT TREEBANK AND RECURSIVE DEEP MODELS FOR COMPOSITIONAL SEMANTICS IN TURKISH

This section introduces the Recursive Neural Network structures and a Morphologically Enriched Turkish Sentiment Treebanks (MS-TR) to operate on them for compositional sentiment analysis of Turkish. Throughout the thesis the abbreviation Tree-RNNs was used for Recursive Neural Networks to prevent mixing with the abbreviation of RNNs for Recurrent Neural Networks.

In this section, we seek to address three main issues for Turkish SA; i) to investigate the effectiveness of the recursive compositional models for Turkish sentiment analysis, ii) to construct a Turkish Sentiment Treebank (a hierarchical representation of the sentences, i.e. fully labelled parse trees) to capture the semantic compositionality in a given sentence, and iii) to contribute to the lack of sentiment analysis resources that also can be used for the other recursive deep models for future studies.

To this end, we introduced our contributions and efforts to develop a novel Morphologically Enriched Turkish Sentiment Treebank (MS-TR) [90], which is annotated according to each word's morphological information. To the best of our knowledge, MS-TR is the first sentiment treebank that is fully labelled according to morphological structures of the Turkish words. It has been constructed to employ compositional effects of the sentiment analysis for two different subtasks: binary classification and fine-grained classification. MS-TR is constructed by using different domains, including movie reviews and multi-domain product reviews, which were annotated by Demirtas and Pechenizky [91]. The fine-grained MS-TR is constructed based on the latest updated version of BOUN Treebank [92]. It was constructed by phrase-level annotation based on the morphological information of each word. The aim is investigating whether recursive neural networks can improve the classification accuracy of MRL. Specifically, we employ Recursive Neu-

ral Tensor Networks (RNTN) for binary and fine-grained sentiment classification over the MS-TR, since RNTN has been performed better compared to the other recursive models such as semi-supervised recursive autoencoders (RAE), matrix-vector recursive neural networks (MV-RNN) both for English [2] and for Arabic, that is also MRL like Turkish [93].

3.1 Preliminaries

Sentences are the hierarchical structures [94]. This means sentences are constructed by combining phrases which are constructed by combining words recursively to represent whole sentence [95].

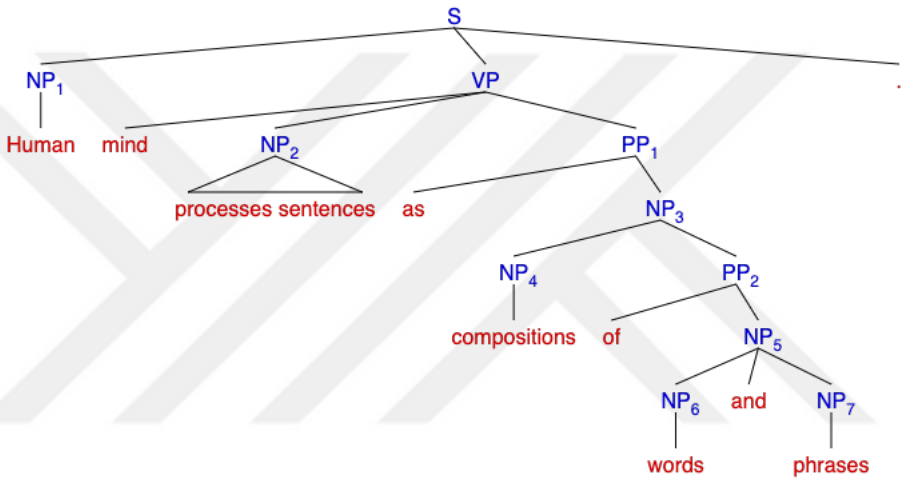

Figure 3.1 An example of hierarchical parse tree of sentences which is constructed recursively by composing noun phrases (NP), verb phrases (VP) and propositions (PP) of the given sentence

The simplest way to model data is representing it as a simple sequence. However, this way cannot capture the structural information encoded in real-world applications data such as protein representations in molecular biology, sentences in natural language processing, or images of computer vision tasks. Learning information about the real-world applications is required to represent data in a structured way, i.e. tree-based or graph-based representation of data. Recursive Neural Networks (Tree-RNNs) were implemented to process the information in a structured network to capture the dynamical connections of complex structures.

While Recurrent Neural Network (RNN) is operating on the chain structures of the sequential input, Recursive Neural Networks (Tree-RNNs) are operating the hierarchical structures, i.e. tree structures to learn compositional features of the data recursively by using tree-based data structures. In the following section details of the Tree-RNNs are given.

3.2 Recursive Neural Networks

Recursive Neural Networks (Tree-RNNs) are inspired by the composition ability of human mind to model the semantic and syntactic structures by tree-like data structures. Tree-RNNs have been designed to learn compositional meanings from hierarchical representations of the sequential data as a generilazed version of Recurrent Neural Networks [27].

The first well-known version of Tree-RNNs was proposed by Kühler and Goller in 1996 [96]. They offer magical powers in many NLP tasks since they can able to learn the structure of language in a recursive essence thanks to bottom up combinations of words by parse tree. They have been used both the exploitation of syntactic information and exploration of semantic information coming from child nodes to the parent nodes for the compositional meaning understanding. Figure 3.2 represents the example parse tree of a given sentence. This is a hierarchical representation of the sentence (top-level node) constructed by the combination of phrases. Every two or more child nodes of parse tree are composed to get high-level compositional phrases until to reach root node which is represented as S in Figure 3.3.

Pollack implemented Recursive Auto-Associative Memory (RAAM) to represent dynamic tree structures as a fixed-length distributional vectors [97]. RAAM architecture is an autoencoder in essence that encodes the leaves of a binary tree in a recursive manner until reach the root of the tree.

Despite every representation models have its advantages and disadvantages [34] using the hierarchical data structures instead of chain structures has many advantages. Tree-RNNs convert chain networks to the deep tree structures. When a same sentence is used to process in both RNN and Tree-RNN, the number of nonlinear compositional function operations is reduced from n to $O(logn)$ in Tree-RNN (here n is defined as sentence (sequence) length). This is the one of the most important advantage of Tree-RNN for dealing with long-term dependencies of RNN. Besides, the representation of text inputs are getting larger by utilising tokens into phrases, phrases into sentences and sentences into documents. Hence it is needed to find more practical compositional models, which can represent larger text units by using fixed dimension. Tree-RNNs combine neighbourhood tokens instead of combining tokens by word orders like in RNN. For example, for the the sentence “something incredible is happening” is processed as a syntactic parse tree, i.e. ((something incredible) (is happening)) over Tree-RNN, instead the sequential input, i.e. (((something incredible) is) happening) chain structure like in traditional RNN.

Traditional RNNs consider only word order in the sentences as a linguistic structure. On the contrary, Tree-RNNs consider the neighbourhood information of the tokens of

sentences and they can learn syntactic and compositional information over parse tree.

Tree-RNNs have produced impressive results for mapping of phrases in a semantic space[98], sentiment analysis [2], dependency parsing [99]. However, the compositional power of the Tree-RNNs is still waiting to explore for Turkish NLP tasks.

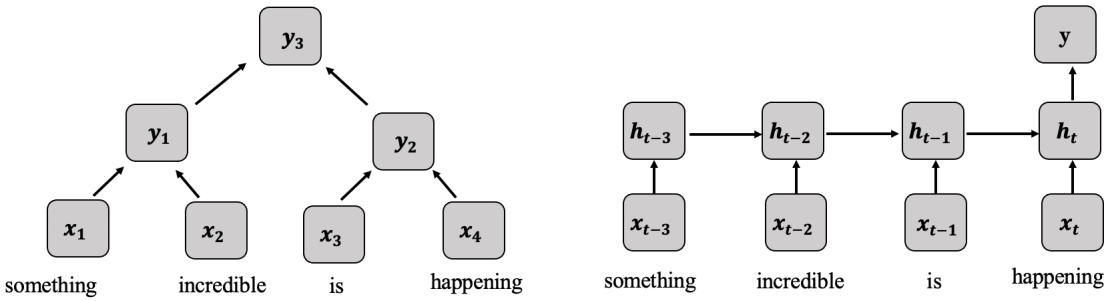

Figure 3.2 Tree-RNN architecture to generate the "something incredible is happening" phrase by composing child nodes recursively on structured parse tree (left), and RNN architecture to model the same phrase by using chain structure that (right)

3.2.1 Recursive Compositional Functions

Bottou pointed out the reasoning ability of the Tree-RNNs [100]. Composition of learnable parameters to achieve a specific task is very important for various NLP tasks. Various versions of Tree-RNNs have been implemented to combine algebraic functions for learning tree-like structures. The main goal is predicting structured representation of sentences from their plausible sub-pieces.

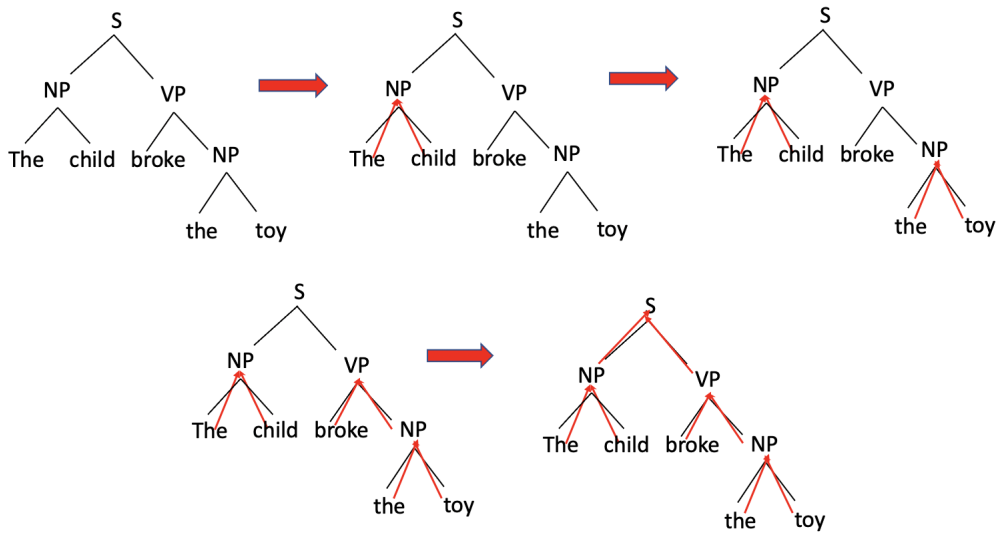

Figure 3.3 Bottom-up recursive tree compositions over a binary parse tree

3.2.1.1 Semi-Supervised Recursive Autoencoders

Autoencoders are unsupervised encoding-decoding networks to learn the approximate representation of an input vector [101–103]. The aim of the autoencoders is learning the lower-dimensional distributional representations of the sparse representations. The input vector x mapped by non-linear function $\psi_e : X \rightarrow \mathscr{F}$ such as $h = \psi_e(x)$ as an encoder and then h is decoded by an output function $\varphi_d : \mathscr{F} \rightarrow X$ such as $y = \varphi_d(h)$. Learning happens by minimizing the following error function:

$$\begin{aligned} L\left(x,\, x\prime\right) &= L\left(x, \varphi_d\left(\psi_e\left(x\right)\right)\right) \\ &= \underset{\varphi_d, \psi_e}{\operatorname{argmin}} \| \, x - \varphi_d\left(h\right) \|^2 = \underset{\varphi_d, \psi_e}{\operatorname{argmin}} \| \, x - x\prime \, \|^2 \end{aligned} \tag{3.1}$$

Here $argmin\| \, x - x\prime \, \|^2$ is defined as reconstruction error using Frobenius norm.

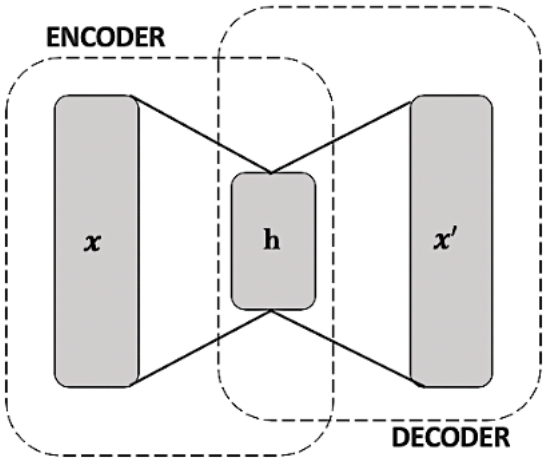

Figure 3.4 Simple architecture of the autoencoder

Recursive Autoencoders (RAE) can learn the hierarchical representation based on the binary tree structures and word embeddings. Assume that $x^{(s)} = (x_1, x_2, \ldots, x_k)$, is the sequence of d-dimensional word vectors of the given sentence for $\forall x_i \in \mathbb{R}^d$, $1 \leq i \leq k$. Figure 3.5 represents the one internal step of the semi-supervised RAE, which is used to map the compositional meanings of the child nodes to the parent vector and to predict the sentiment class of the reconstructed parent representation. As given in Figure 3.17, $x^{(s)} = (x_1, x_2, \ldots, x_k)$ has been learned by using binary tree structure by evaluating parent word vectors with following recursive function:

$$\begin{aligned} x_p &= \varphi\left(W_{emb}\left[x_l; x_r\right],\, b_{emb}\right) \\ &= \varphi\left(W_{emb} \begin{bmatrix} x_l \\ x_r \end{bmatrix},\, b_{emb}\right) \end{aligned} \tag{3.2}$$

Here φ represents the tanh function and x_l, x_r and x_p are $\mathbb{R}^{dx1}$ dimensional word vectors of left child word, right child word and parent word respectively. The concatenation of the x_l, x_r is represented by $[x_l; x_r] \in \mathbb{R}^{2dx1}, W_{emb} \in \mathbb{R}^{dx2d}$ and $b_{emb} \in \mathbb{R}^{dx1}$. The quality of the parent vector is calculated by reconstruction layer as follows:

$$[x_l\prime; x_r\prime] = W_{rec} x_p + b_{rec} \tag{3.3}$$

Here $W_{rec} \in \mathbb{R}^{dx2d}$, $b_{rec} \in \mathbb{R}^{dx1}$ and reconstruction layer is doing similar task to the φ_d as defined in 3.1. The aim is minimizing the reconstruction error, which is defined same such as 3.1:

$$E_{EMB} = L\left([x_l; x_r],\ [x_l\prime; x_r\prime]\right) = argmin \| \ [x_l; x_r] - [x_l\prime; x_r\prime] \ \|^2 \tag{3.4}$$

At every internal the parent word vector x_p scored with positive or negative by using softmax function for binary classification task. The phrases are merged recursively to predict sentiment class of the root vector as follows:

$$\begin{aligned} y^{x_p} &= softmax\left(W_{label} x_p\right) \\ &= \frac{exp(W_{label} x_p)}{\sum_{i=1}^{c} exp(W_{label}\ x_{p_i})} \end{aligned} \tag{3.5}$$

y^{x_p} is the predicted sentiment, $W_{label} \in \mathbb{R}^{cxd}$, and c is the number of sentiment class. Here $y^{x_p} = softmax\left(W_{label} x_p\right)$ can be interpreted as a conditional probability $y^{x_p} = p(c|[x_l; x_r])$ and $\sum_{i=1}^{c} y^{x_{pi}} = 1$ and for a given target (true) label t^{x_p} of the parent vector, the total cross-entropy loss is defined as follows:

$$\begin{aligned} E_{CE} &= -\sum_{i=1}^{c} E(x_{pi}, t^{x_{pi}}, \theta) \\ &= -\sum_{i=1}^{c} t^{x_{pi}} \log(y^{x_{pi}}) \end{aligned} \tag{3.6}$$

The total objective function is defined by weighted sum of the reconstruction error of the compositional embeddings is E_{EMB} and the cross-entropy error of the sentiment labels E_{CE} as follows for a given pair (x^s, t^{x_s}) from the dataset:

$$L_{total} = \frac{1}{N_{out}} \sum_{i=1}^{N_{out}} E(x_i, t^{x_i}, \theta) + \frac{\lambda}{2} \| \ \theta \ \|_2^2 \tag{3.7}$$

the error of each pair (x^s, t^{x^s}) for a given sentence from the train dataset is defined as

follows:

$$E(x^s, t^{x^s}, \theta) = \sum_{i=1}^{BTree \in x^{(s)}} E([x_l; x_r], x_{pi}, t^{x_{pi}}, \theta) \tag{3.8}$$

where BTree is the set of binary trees that contains non-leaf nodes of a given sentences and $E([x_l; x_r], x_{pi}, t^{x_{pi}}, \theta)$ is calculated weighted sum of E_{EMD} and E_{CE}:

$$E([x_l; x_r], x_p, t^{x_p}, \theta) = \sum_{i=1}^{BTree \in x^{(s)}} \alpha E_{EMB}([x_l; x_r]; \theta) + (1-\alpha) E_{CE}(x_p, t^{x_p}, \theta) \tag{3.9}$$

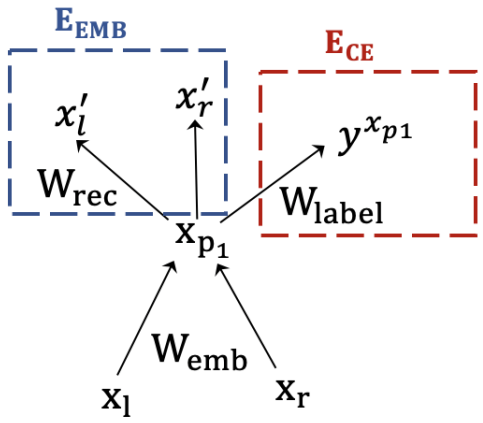

Figure 3.5 Inner node of the recursive autoencoder

The learning happens by updating the learnable parameters $\theta = (W_{emb}, W_{rec}, W_{label}, b_{emb}, b_{rec})$. The gradient is defined as follows:

$$\frac{\partial L_{total}}{\partial \theta} = \frac{1}{N_{out}} \sum_{i=1}^{N_{out}} \frac{\partial E(x_i, t^{x_i}, \theta)}{\partial \theta} + \lambda \theta \tag{3.10}$$

The parameter updating happens based on the gradient descent learning rule by backpropagation through structure [96].

3.2.2 Learning Through Structure

In this section the learning procedure of the tree-structured networks was explained. Backpropagation through structure (BPTS) is used to minimize cross-entropy error.

Principally, BPTS learning procedure follows the similar way as BPTT. Updating of the learnable parameters happens by backpropagation through structure and the derivative of

each parent node through the tree structure is needed to be calculated. To this end, as Goller and Kuchler proposed in 1996 [96] BPTS follows following pathways:

(i) At every parent nodes of the tree structure the error need to be splitted to its children. The predicted sentiment y^{x_p} (3.23) of the parent vector x_p (3.17) is used to minimize the cross-entropy error message δ^{x_p} based on the ground truth value t^{x_p}:

$$\delta^{x_p} = ({W_{label}}^T (y^{x_p} - t^{x_p})) \otimes \phi'(x_p) \tag{3.11}$$

The negative log-likelihood loss function, that is defined in 3.24 is used to minimize the cross-entropy. y^{x_p} is the predicted sentiment of the parent vector x_p are $\mathbb{R}^{dx1}$.

$$\begin{aligned} \delta^{x_p,down} &= ({W_{emb}}^T \delta^{x_p}) \otimes \varphi'(x_p) \\ &= ({W_{emb}}^T \delta^{x_p}) \otimes \varphi' \begin{bmatrix} x_l \\ x_r \end{bmatrix} \\ \delta^{x_l} &= \delta^{x_p,down}[1:d] \\ \delta^{x_r} &= \delta^{x_p,down}[d+1:2d] \end{aligned} \tag{3.12}$$

If for example the right-child has a label the error function of it will be like the following:

$$\delta^{x_r,combined} = \delta^{x_r,,softmax} + \delta^{x_p,down}[d+1:2d] \tag{3.13}$$

(ii) The summation of the partial errors over tree structure needed to be calculated. To illustrate the recursive derivative function with an example, the derivative of the recursive function that is defined as $f_{rec} = \varphi(W_{emb}(\varphi(W_{emb}x_p)))$ can be calculated according to the W_{emb} as follows:

$$\begin{aligned} \frac{\partial f_{rec}}{\partial W_{emb}} &= \frac{\partial \left[\varphi(W_{emb}(\varphi(W_{emb}x_p)))\right]}{\partial W_{emb}} \\ &= \varphi'(W_{emb}(\varphi(W_{emb}x_p))) \left[\frac{\partial W_{emb}}{\partial W_{emb}}\varphi(W_{emb}x_p) + W_{emb}\frac{\partial \varphi(W_{emb}x_p)}{\partial W_{emb}}\right] \\ &= \varphi'(W_{emb}(\varphi(W_{emb}x_p))) \left[1.\varphi(W_{emb}x_p) + W_{emb}\varphi'(W_{emb}.x_p)x_p\right] \end{aligned} \tag{3.14}$$

This equation gives the same result with each embedding matrix if the function defined as $f_{rec} = \varphi(W_{E_2}(\varphi(W_{E_1}x_p)))$ and the derivative is calculated as follows:

$$
\begin{aligned}
&= \frac{\partial\left[\varphi(W_{E_2}(\varphi(W_{E_1}x_p)))\right]}{\partial W_{E_2}} + \frac{\partial\left[\varphi(W_{E_2}(\varphi(W_{E_1}x_p)))\right]}{\partial W_{E_1}} \\
&= \varphi'(W_{E_2}(\varphi(W_{E_1}x_p)))(\varphi(W_{E_1}x_p)) + \varphi'(W_{E_2}(\varphi(W_{E_1}x_p)))(W_{E_2}\varphi'(W_{E_1}x_p)x_p) \\
&= \varphi'(W_{E_2}(\varphi(W_{E_1}x_p)))\left[\varphi(W_{E_1}x_p) + W_{E_2}\varphi'(W_{E_1}x_p)x_p)\right]
\end{aligned}
\tag{3.15}
$$

If $W_{E_2} = W_{E_1}$ it is clear that Equation 3.15 is equal to the Equation 3.14.

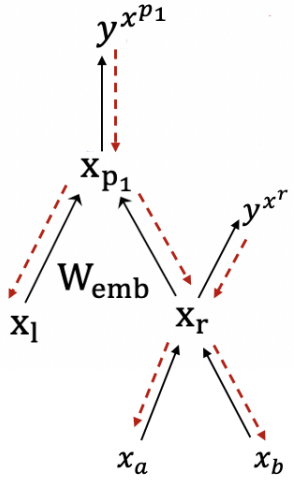

Figure 3.6 Splitting error function through structure and backpropagation of the error

Tree-RNNs have been developed by Socher *et al.* [2] to model meaning in long-phrases and sentences by a recursive combination functions in a neural network's architecture. Tree-RNNs are needed to be specialised labeled tree structures to train for sentiment analysis task. Stanford Sentiment Treebank (SST) had been developed from the dataset English classification datasets (fine-grained) [104], and Tree-RNNs have been achieved impressive successes for sentiment analysis in English by using SST.

The main difficulty in terms of the semantics is to capture the meaning of longer phrases. This section mainly focused on combining word-phrase level and sentence-level sentiment information, with the implementations of various datasets from different domains, including movie reviews and various product reviews. Most of the previous studies have been done based on a binary prediction of a given sentence due to limited resources of a fine-grained labelled large dataset. Hence, a novel fine-grained sentiment classification dataset has been also introduced to contribute to the lack of Turkish NLP resources.

3.3 MS-TR: A Morphologically Enriched Sentiment Treebank for Compositional Semantics

In this section, we introduce MS-TR, to address compositional sentiment analysis for Turkish. We propose a semi-supervised automatic annotation, as a distant-supervision approach, using morphological features of words to infer the polarity of the inner nodes of MS-TR as a positive and negative. The proposed annotation model has been done for 4 levels, including morphemes, stems, tokens, and reviews. Each annotation level's contribution was tested using three different domain datasets, including product reviews, movie reviews, and the Turkish Natural Corpus essays. Comparative results were obtained with the RNTNs which is operated over MS-TR, and conventional ML methods. Experiments proved that RNTN outperformed the baseline methods and achieved much better accuracy results compare to the baseline methods, which cannot accurately capture the aggregated sentiment information. The following sections presents the proposed methodologies to construct fully-labelled MS-TR.

3.3.1 System Architecture, Resources and Tools Used in Building MS-TR

MS-TR has been constructed as a new sentiment analysis resource to work with tree-structured Recursive Deep Learning models. The main drawback for the data-driven classification systems is to have task-dependent well constructed annotated data. Researchers have proposed to use emoticons, and emojis [105, 106] or hashtags [107] as classification labels inferring polarity of a given task as an extension of distant supervision [108] to overcome this issue. Similar to those approaches, we propose using morphological features of each word to infer the polarity of the inner nodes of the MS-TR, which contains annotated binary-structured parse trees. Parsed trees were labelled according to fine-grained (multi-class) and binary sentiment class.

Datasets Two different dataset format have been used for the construction of the MS-TR. Binary-labelled MS-TR was constructed using movie reviews and multi-domain product reviews with a raw text format. A fully-labelled fine-grained MS-TR was constructed using the latest updated version of BOUN Treebank [92]. BOUN Treebank includes 9,757 sentences from different application domains such as newspapers, magazines and essays of Turkish Natural Corpus [109]. Each sentence in BOUN Treebank encoded in ConLL-U format. Movie reviews collected by Demirtas and Pechenizkiy [91] from beyazperde.com. Multi-domain product reviews include reviews about the books, DVD, etc. that are collected from Turkish commercial website hepsiburada.com. Additional details and the average and maximum n-grams lengths of reviews are given in Table 3.1 and Table 3.2.

Table 3.1 Maximum and average n-grams length of the products datasets of MS-TR

Datasets	Max N-Gram Length	Avr. N-Gram Length
Books	207	33.19
DVD	253	31.75
Electronic	245	37.51
Kitchen	182	32.66
Movie	1566	33.20

Table 3.2 Balanced Turkish multi-domain products movie reviews datasets

TR Books	700 P, 700 N
TR DVD	700 P, 700 N
TR Electronics	700 P, 700 N
TR Kitchen Appliances	700 P, 700 N
Turkish Movie Reviews	5331 P, 5331 N

Figure 3.8 illustrates steps for the annotation steps of MS-TR and binarization process. The proposed pipeline starts with preprocessing steps and sentence boundary detection. Before the annotation and morphological analysis, each review in the dataset is parsed at the sentence boundary detection phase and then cleaned by Turkish spell checking. Zemberek was used [110] for preprocessing steps, including sentence boundary detection and morphological parser. Turkish spell checking also had been done since Turkish has additional letters such as (“ç”, “ğ”, “ı”, “ö”, “ş”, “ü”), that is missing in English, the raw text from the dataset usually contains non-Turkish characters because of the non-Turkish keyboards. Each word morphologically analyzed and labelled according to the negation cases, as discussed in the previous section. Additionally, we used a polar word list to detect sentiment specific adjectives and nouns in an unsupervised manner. Zemberek is only used for the movie reviews, and product reviews as a morphological analyzer since CoNLL-U format dataset has already contained FEATS field that contains morphological features for words.

In Figure 3.7, it can be seen the parse tree of two different sentences: “Bu bağlamda inceleyeceğimiz iki gök haritası somut örnekler sunacak.” and “Zeki Müren , Müzeyyen Senar , Safiye Ayla çocukluğumdan beri tanış olduğum isimler.” from the CoNNL-U dataset. The word “sunacak” is a root of the parse tree, and it can be seen POS tags (i.e. NOUN, ADJ, PUNCT) of the leaf words and root word of the parse tree. These parse trees (graphs) are examples of structural representation of the sentences that can be processed in Recursive Neural Models (Tree-RNNs).

Tree-RNNs were proposed for learning from the arbitrary shape structures like trees and graphs; hence it is needed to implement a binary parse tree to train RNTNs. The Stanford Core NLP tool has been used to build the binarized dataset, i.e. fully labelled parse trees

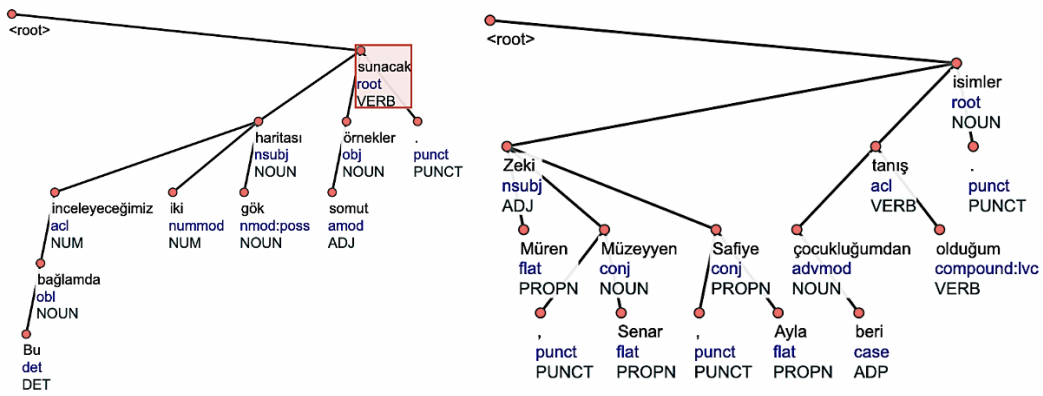

Figure 3.7 A constituency parse tree of two different sentences from the BOUN treebank [92]

at the final step of the MS-TR construction. This a tool to set the class labels on a sentence tree dataset using the user default annotations. The expected input file for the Stanford Core NLP should contain one sentence per line. The reviews separated by blank lines and each review labelled with their sentiments. Before the blank line, sub-phrases of the current review can also be labelled with their own label. Consequently, all of the labels on a tree set to the given default value. The annotation strategies of the MS-TR is different from the Stanford Sentiment Treebank (STS), which is developed by Socher *et al.* [2] for English. We propose to annotate MS-TR in a semi-supervised manner to construct fully-labelled parse trees. The following section gives the details and pseudo-codes of the proposed annotation phase of the MS-TR.

3.3.2 Semi-Supervised Annotation Strategies of the Turkish Sentiment Treebank

We present two pathway as an automatic semi-supervised annotation approach for binary classification datasets. The first one is using the combination of morphological features of words, and the other is hybridization of the polarity lexicon and the polar word embedding models. By following two pathways, we have constructed four different treebanks, namely morph-level (stems+suffixes) annotated MS-TR, stem-level annotated MS-TR, token-level annotated MS-TR, and review-level annotated MS-TR. Each model has been constructed for each dataset to figure out the compositional effects of the morphologically rich structure of Turkish. Each annotation level has been tested to compare the efficiency of the proposed models by feeding them into the RNTNs.

3.3.3 Morphological Analysis of Words for Annotation

Morphological features of the Turkish words are very important for accurate sentiment analysis since even suffixes of the Turkish words can have the polarity information as

discussed in above. Particularly, negation suffixes, that are embedded within the word are crucial for the detection of the negative polarity. We annotated MS-TR by using the morphological features of the words to explore sentiment that hides behind Inflectional Groups (IG) of words. We parse words to the root of the word and the possible suffixes to find morphological features. We consider the following cases for a binary level annotation:

Case 1: In Turkish, the suffix -mA and its versions are used to negate the verbs. For example, the suffix -me changes the sentiment of word from positive meaning such as "sevdim" (I like it) to the negative such as "sevmedim" (I did not like it). In addition, Turkish auxiliary verbs "etmek" (to do, to make) and "olmak" (to be, to become) are used with the suffix -mA to negate the verb phrases. For example, "sabretmemek" was annotated as negative since it is morphological features sabret+Verb+**Neg**^DB+Noun+Inf1+A3sg contains "Neg" keyword (see Table 3.3).

Table 3.3 Examples of the morphological analysis of words and detecting negation within words

(a) Word / Phrases **Possible Morphological Analysis**
sabretmek (to be patient) sabret+Verb^DB+Noun+Inf1+A3sg
sabretmemek (not to be patient) sabret+Verb+**Neg**^DB+Noun+Inf1+A3sg
sabırlı (patient) sabırlı+Noun+Prop+A3sg sabır+Noun+A3sg^DB+Adj+**With** sabır+Noun+Prop+A3sg^DB+Adj+**With**
sabırsız (impatient) sabır+Noun+A3sg^DB+Adj+**Without** sabır+Noun+Prop+A3sg^DB+Adj+**Without**
sabırlı değil(without patient) değil+Conj değil+Verb+**Neg**+Pres+A3sg
sabrı **yok** (have no patience) yok+Conj yok+Adj yok+Noun+Prop+A3sg
sabrı **var** (does have a patience) var+Adj var+Verb+Imp+A2sg var+Noun+A3sg

Case 2: Besides verbal negation, the adjectives, which are transformed from the noun using absence suffixes (-lı/-li) and presence suffixes (-sız/siz) can give polarity infor-

Table 3.4 Examples of the polar words which don't have polar suffixes

Word / Phrases **Polarity (P:Positive N:Negative)**
kazanan (winner) (P)
kaybeden (loser) (N)
asimile (assimilated) (N)
karizmatik (charismatic) (P)
dağınık, savruk (untidy) (N)
muhteşem, harika (gorgeous) (P)
üçkağıt (fiddle) (N)
dingin (calm) (P
galeyan (ebullition) (N)
filizlenmek (sprout) (P)
pandemi, salgın (pandemic, epidemic) (N)
iyi (good) (P)
çene çalmak, gevezelik (chitchat, chatter) (N)
nezaket, nazik (kindness, gentle) (P)
kötümser, karamsar (pessimistic) (N)
efsanevi (legendary) (P)
antisosyal (antisocial) (N)
öncü (pioneer) (P)
sendrom (syndrome) (N)
alçakgönüllü (humble) (P)
intihal (plagiarism) (N)
aklıselim (common sense) (P)

mation. The suffix -lı and its versions transform names to the positive adjectives, and the suffix -sız and its versions transform names to the negative adjectives. For example, the name "sabır" transforms to the positive adjective with the suffix -lı, such as "sabırlı" (patient). The suffix -sız (without) negates adjectives and converts its positive meaning to the negative meaning, such as "sabırsız" (impatient). We detect the absence and presence by considering the keywords (Without/With) in the morphological analysis of the words. Since one of the morphological analysis of "sabırlı" (sabır+Noun+A3sgˆDB+Adj+With) contains the keyword "With", sabırlı annotated as positive. Similarly, "sabırsız" is annotated as negative since its morphological analysis contains "Without" (sabır+Noun+A3sgˆDB+Adj+Without).

Case 3: The conjunction "değil" (is/are not) is used with adjectives as a negation marker. For example, "uzun değil" means "not long" gives a negative meaning to the "uzun" (long) as an adjective. Similarly, "var" (there is) and "yok" (there is not) are also used as a presence and absence indicator for nouns. For example, "sabrı yok" (have no patience) has negative polarity, and "sabrı var" (have patience) has a positive polarity. We annotated "değil and "yok" with as a negative; and "var" as a positive.

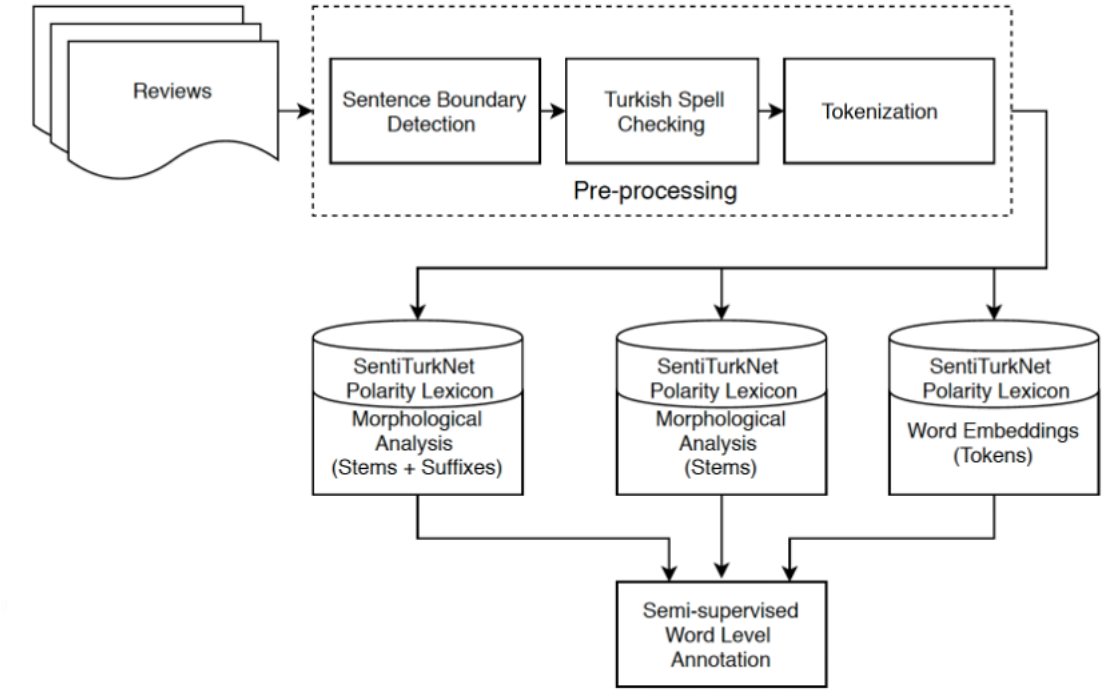

Figure 3.8 The pipeline of the binarization framework to construct MS-TR

3.3.3.1 Morph-Level Annotated MS-TR

In this model, the annotation has been done at morph-level, aiming to retrieve hidden polarity of the suffixes in the morphologically rich word. We propose to construct fully-labelled morphologically enriched treebank (Morph-Level MS-TR) with the morphological features of the words, which are used as a distant supervision method similar to the [105–108]. To this end, tokens are morphologically analyzed, and they are parsed to their possible stem and suffixes. All of the possible morphological results for a given input word were used for the word-level annotation. If the word contain negation suffixes as given in previous section, the parsed stem and suffixes (ending) of the word labelled with 0, if it contains positive suffixes the parsed stem and suffixes (ending) of the word have been annotated with 1. For example, sabırsız (sabır+sız) "impatient" is represented as ((0 sabır) (0 sız)) in the proposed Morph-Level Annotated MS-TR. Similarly, sabırlı (sabır+lı) "patient" is represented as ((1 sabır)(1 lı)) as a morph-level annotated tree in the MS-TR.

In addition to the morphological annotation, we propose to use polarity lexicon to capture polar words, which are root and do not have any morphological information. Words in Table 3.4 have been used as a case examples to elaborate on this point. Using only a morphological analysis of the word cannot provide the correct polarity information for each case. For example, the sentence "Seni yeniden görmek için sabırsızlanıyorum" (I can't wait to see you again / looking forward to seeing you again) has a positive meaning, but the morphological information of the "sabırsızlanıyorum" contain negation suffix -

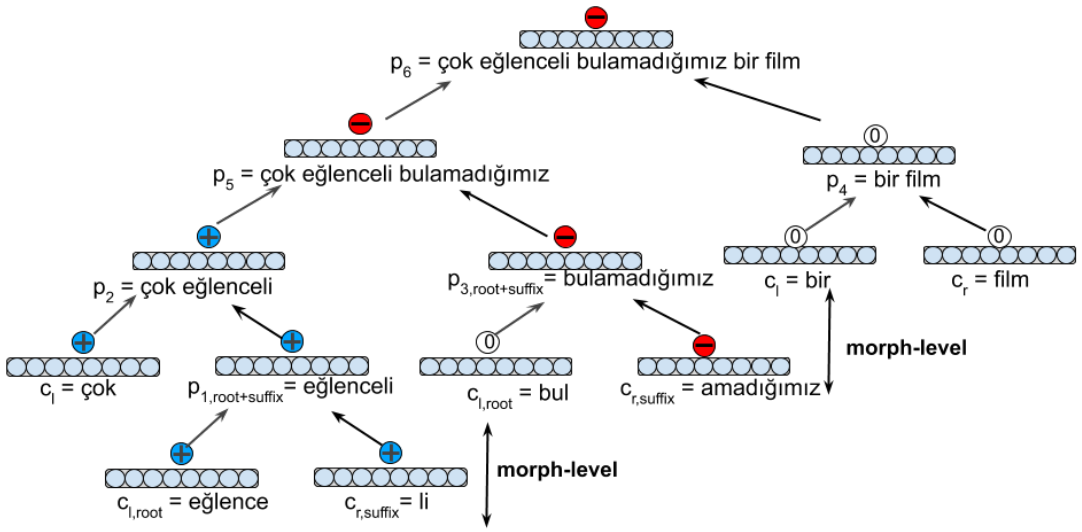

Figure 3.9 An example for the morph-level annotated tree structure of phrase çok eğlenceli bulamadığımız bir film, “a movie that we could not find much enjoyable” from MS-TR

sız (sabır+sızlanıyorum). To handle this issue, we proposed to use sentiment polarity lexicon, SentiTurkNet [10], and we combine morphological features and polarity lexicon information to annotate each word of the review. After the tokenization step, the polarity words are controlled by the hybrid usage of morphological analysis and SentiTurkNet lexicon. This approach used for binary annotation of the MS-TR. The detailed steps of the annotation algorithm are given in Algorithm 7.

Figure 3.9 represents the hierarchical structure of the phrase *çok eğlenceli bulamadığımız bir film*, “a movie that we could not find much enjoyable". Each token of the phrase has been parsed and annotated to learn review level sentiment recursively in a bottom-up manner. As it can be seen at the morph-level, suffix –li “with” and root word eğlence “enjoy” are composed as a distributional word vectors of left child $(c_{l,root})$ and right child $(c_{r,suffix})$ to calculate parent representation of the word eğlenceli “enjoyable”. This combination combines the positive sentiments from child nodes to the parent node. The second morph-level composition is a clear example of the handling negation of the morphologically rich word. The word bulamadığımız “that we could not find”, contains a negation suffix -ma ”without”, hence the right child $(c_{r,suffix})$ is negative. As a result the compositional parent node bulamadığımız “that we did not find” is also negative. The longest phrase *çok eğlenceli bulamadığımız bir film*, “a movie that we could not find much enjoyable” is produced recursively at the same dimension in a bottom-up manner and annotated as a negative.

Algorithm 7: Morph-level semi-supervised binary annotation

Input: LabelledReviews, Polarity Lexicon
Output: Morph-Level Annotated Parsed Reviews
Function `MorphToLabel` (*LabelledReviews, polarityLexicon*)

```
seperatedReviews = SentenceBoundaryDetection(labelledReviews)
cleanedReviews = SpellChecking(seperatedReviews)
tokenizedReviews = Tokenization(cleanedReviews)
annotatedFile = openNewFile4AnnotatedReviews()
TRMorphology = Zemberek.TurkishMorphologyAnalyzer()
for each review ∈ tokenizedReviews do
    annotatedFile.write(review)
    annotatedTokens= [ ]
    for each token ∈ review do
        tokenMorphology = TRMorphology(token)
        stem = tokenMorphology.getStem()
        suffix = tokenMorphology.getSuffix()
        // Negation Handling
        if tokenMorphology contains "Neg"or"Without" then
            labelledMorph = 0 + stem + 0 + suffix
            annotatedTokens.add(labelledMorph)
            break
        else if tokenMorphology contains "Pos"or"With" then
            labelledMorph = 1 + stem + 1 + suffix
            annotatedTokens.add(labelledMorph)
            break
        // Annotation with Polarity Lexicon
        PosPolarWords = getPositivesFromLex(SentiTurkNet)
        NegPolarWords = getNegativesFromLex(SentiTurkNet)
        if PosPolarWords contains token then
            labelledMorph = 1 + token
            annotatedTokens.add(labelledMorph)
            break
        else if PosPolarWords contains stem then
            labelledMorph = 1 + stem
            annotatedTokens.add(labelledMorph)
            break
        if NegPolarWords contains token then
            labelledMorph = 0 + token
            annotatedTokens.add(labelledMorph)
            break
        else if NegPolarWords contains stem then
            labelledMorph = 0 + stem
            annotatedTokens.add(labelledMorph)
            break
    annotatedFile.write(annotatedTokens)
return annotatedFile
```

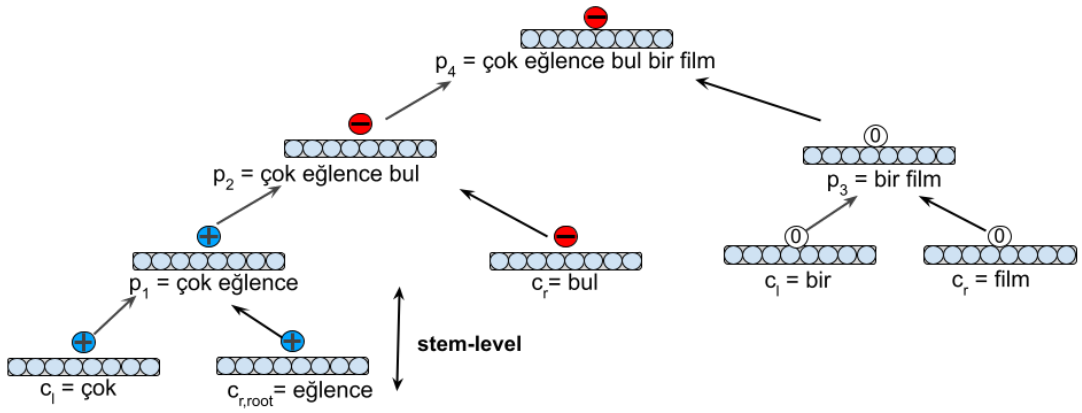

Figure 3.10 An example representation for the stem-level annotated tree structure of phrase *çok eğlenceli bulamadığımız bir film*, "a movie that we could not find much enjoyable" from MS-TR

3.3.3.2 Stem-Level Annotated MS-TR

Stem-Level annotated MS-TR was constructed similar to morph-level annotated MS-TR. The only difference is using only stems of the words, and the suffixes (ending) of the words are eliminated from word after morphological parsing. The aim of constructing stem-level MS-TR is to investigate the efficiency of the using stem+suffix structure by comparing to using the only stem of the words. The flowchart of the annotation algorithm is similar to Algorithm 7 and the example tree structure of the stem-level annotated MS-TR is given in Figure 3.10.

3.3.3.3 Token-Level (Surface Level) Annotated MS-TR

As a third level, we propose to annotate each token of the reviews using polar embedding spaces, which are constructed by using positive and negative datasets. Related word vectors have been produced for each polarity level using the word embeddings model. FastText [111] model has been used to take its advantage of representing out-of-vocabulary word vectors for MRLs. To this end, the positive embedding model has been constructed by using positive reviews dataset and positive polar words that are taken from SentiTurkNet. Similarly, negative word embedding space has been constructed using negative reviews dataset and negative polar words taken from SentiTurkNet. The most similar word of the token has been found by using positive word vector space and negative word vector space.

After tokenization step of each review, the cosine similarity measure has been used to find the token-level label. If the cosine similarity of the positive most similar word and the target token is bigger than the cosine similarity of the negative most similar word and the target token, the token is labelled as positive; else it is labelled as negative. The flowchart of the proposed algorithm is given in Algorithm 8, and the example tree structure of the

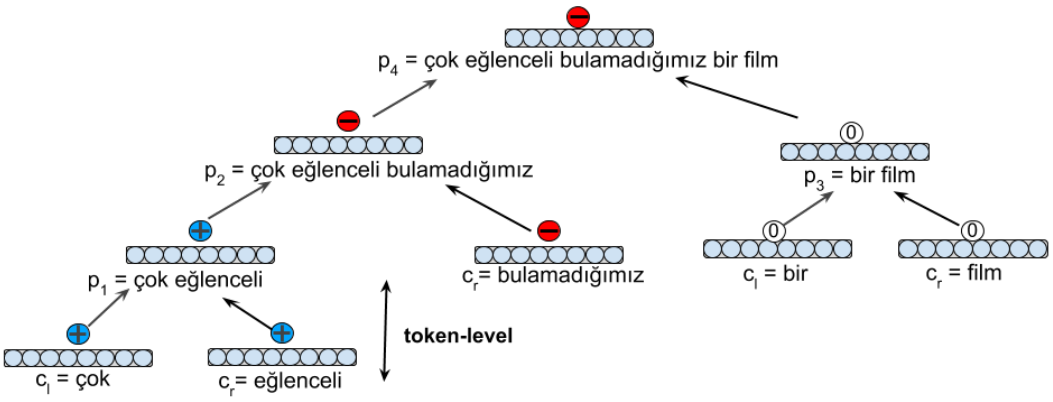

Figure 3.11 An example representation for the token-level annotated tree structure of phrase *çok eğlenceli bulamadığımız bir film*, "a movie that we could not find much enjoyable" from MS-TR

token-level annotated MS-TR is given in Figure 3.11.

3.3.3.4 Review-Level Annotated MS-TR

As the last annotation level, we proposed using only review-level annotated tree structures to construct MS-TR for comparing each annotation level's performance. After the annotation process has been done, the annotated files fed into the Standford Core NLP module to construct the binarized tree structure by parsing each review of the datasets.

In addition to the binary-labelled MS-TRs, fine-grained MS-TR has been annotated by taking into account the morphological information of the words. 1003 of BOUN Treebank sentences have been selected randomly as an initial step. Parsing the 1003 sentences has produced 63,782 nodes (see Table 3.5). The polarity distribution of the phrases, i.e. phrase-level to review-level labelling was realized in two stages. The first stage is labelling words according to their polarity features which are detected from the morphological analysis of the words. The morphological feature of the word was detected from the FEATS field of the CoNLL-U data format, which provides the lemma and morphological analysis of the word, including Polarity feature as proposed in the previous section. In addition to the binary annotation, we have scored words whether they are booster words. If the word is contained by positive booster words list, it was annotated with 5 for very positive class. Similarly, if the negative booster words list contains the word, it was annotated with 1 for very negative class. The total sentence-level score was calculated as follows:

$$senti_{Score} = -10x_{veryNeg} - 5x_{neg} + 5x_{pos} + 10x_{veryPos} \tag{3.16}$$

Here each x represents the total number of related polar word in a given sentence. Booster

Algorithm 8: Semi-Supervised Binary Annotation With Word Embeddings Model

Input: LabelledReviews, Polarity Lexicon
Output: Token-Level Annotated Parsed Reviews
Function `MorphToLabel` (*LabelledReviews, polarityLexicon*)

```
Start
posReviews = GetPositiveReviews(labelledReviews)
negReviews = GetNegativeReviews(labelledReviews)
posPolarWords = GetPosPolarWords(SentiTurkNet)
negPolarWords = GetNegPolarWords(SentiTurkNet)
annotatedFile = openNewFile4AnnotatedReviews
PosVocabulary = posReviews + posPolarWords
NegVocabulary = negReviews + negPolarWords
// Word Embedding Models with FastText
PosModel = FastText.train(PosVocabulary)
NegModel = FastText.train(NegVocabulary)
for each review ∈ LabelledReviews do
    annotatedFile.write(review)
    annotatedTokens= [ ]
    for each token ∈ review do
        posSimilarity = PosModel.GetMostSimilar(token)
        negSimilarity = NegModel.GetMostSimilar(token)
        // Annotate According to Cosine Word
         Similarity
        if posSimilarity geater or equal to negSimilarity then
            labelledToken = 1 + token
            annotatedTokens.add(labelledToken)
            break
        else
            labelledToken = 0 + token
            annotatedTokens.add(labelledToken)
            break
    annotatedFile.write(annotatedTokens)
return annotatedFile
```

words were taken from polarity lexicon, which are used for morph-level and stem-level annotation. After calculating the total sentiment score of the sentence, scores of -15 and less are labelled as very negative (1), those with -10 labelled as negative (2), 0 is labelled as neutral (3), 10 is labelled as positive (4), and those above 15 and 15 are labelled as very positive (5). Table 3.5 represents the total number of the fine-grained n-grams for each sentiment class. Somewhat positive and somewhat negative n-grams got emotion scores of 5 and -5, respectively, hence we considered them in positive and negative, respectively. After semi-supervised annotation process, 188 reviews were labelled as class 5 (very positive), 215 reviews were labelled as positive, 230 reviews were labelled as neutral, 207 reviews labelled as negative, and 163 reviews labelled as very negative.

Table 3.5 Total numbers of fine-grained n-grams

N-Grams	Total Number
Very Positive	8231
Positive	7610
Somewhat Positive	22635
Notr	15392
Somewhat Negative	2471
Negative	2559
Very Negative	214

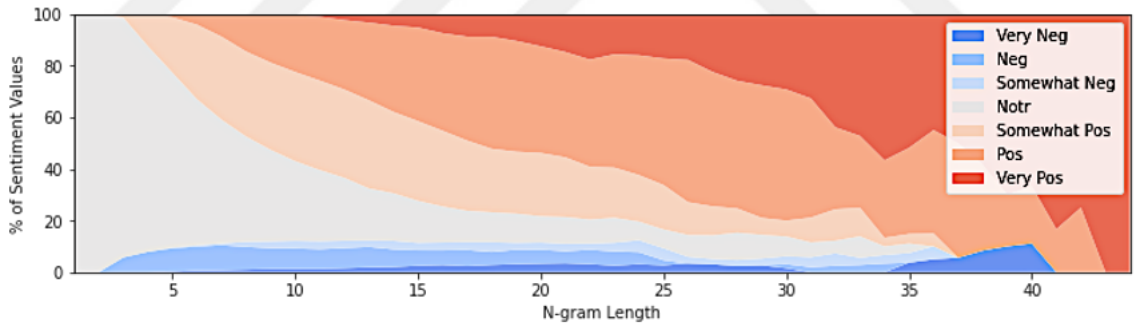

Figure 3.12 Normalized histogram of the annotated n-grams in fine-grained MS-TR. Somewhat positive distributions are added to the positive sentiment class, and somewhat negative distributions are added to the negative sentiment class for fine-grained sentiment classification

We used the Standford Core NLP [112] module to build a binarized tree structure similar to the Stanford Sentiment Treebank (STS) [113]. The total inner nodes of the annotated MS-TR that have been produced by binarized parsing have been given in 3.5.

3.4 Recursive Deep Models over MS-TR for Compositional Semantics

The compositional power of the Tree-RNNs is still waiting to explore for Turkish sentiment analysis task. With this motivation, we employ MS-TR with Recursive Neural Tensor Networks (RNTN) to handle Turkish's agglutinative morphology and catch the freedom of its constituent structure for compositional sentiment analysis. The proposed model also contributes to the lack of data sources for improving understanding of semantics in Turkish.

Recursive Neural Network (Tree-RNN) is a tree-structured model based on composing words over nested hierarchical structure in sentences. The neural network function recursively merges words to construct noun phrases until representing the entire sentence. Tree-RNNs have an extraordinary ability for mapping of phrases in a semantic space [98].

Recursive Neural Tensor Networks (RNTN) have achieved promising accuracy results for English [2] and morphologically rich languages (MRL) such as Arabic [3]. RNTN outperformed the previous versions of the Recursive Neural Networks, such as Tree-RNNs and MV-RNN. Hence we used RNTN over a morphologically enriched Turkish sentiment treebank.

The proposed model learns the distributional representations to construct not only phrases to but also morphologically rich words from their root and suffixes.

Tree-RNNs are the simplest version of the Recursive Deep Models. In essence, they are designed to process tree-structured datasets, and they generalize the sequential models from chain-structures to the tree-structures. Assume that $x^{(s)} = (x_1, x_2, \ldots, x_k)$, is the sequence of d-dimensional word vectors of the given sentence for $\forall x_i \in \mathbb{R}^d$, $1 \leq i \leq k$. $x^{(s)} = (x_1, x_2, \ldots, x_k)$ has been learned by using binary tree structure by evaluating parent word vectors with following recursive function:

$$\begin{aligned} x_p &= \varphi\ (W_{emb}\ [x_l; x_r],\ b_{emb}) \\ &= \varphi\ (W_{emb} \begin{bmatrix} x_l \\ x_r \end{bmatrix},\ b_{emb}) \end{aligned} \tag{3.17}$$

Here φ represents the tanh function and x_l, x_r and x_p are $\mathbb{R}^{dx1}$ dimensional word vectors. The concatenation of the x_l, x_r is represented by $[x_l; x_r] \in \mathbb{R}^{2dx1}, W_{emb} \in \mathbb{R}^{dx2d}$ and $b_{emb} \in \mathbb{R}^{dx1}$.

RNTNs is the enhanced version of the tree-based deep learning models, which aims to aggregate polarity of child nodes to the parent root node using one direct composing

relation by tensor algebra [2]. The learning architecture of the model is the same as the Tree-RNN. The modification has been done using the following aggregation:

$$h_{pi} = \begin{bmatrix} x_l \\ x_r \end{bmatrix}^{\mathsf{T}} T^{[i]} \begin{bmatrix} x_l \\ x_r \end{bmatrix} \tag{3.18}$$

$$h_p = \begin{bmatrix} x_l \\ x_r \end{bmatrix}^{\mathsf{T}} T^{[1:d]} \begin{bmatrix} x_l \\ x_r \end{bmatrix} + W_{emb} \begin{bmatrix} x_l \\ x_r \end{bmatrix} \tag{3.19}$$

$$x_p = \varphi\left(\begin{bmatrix} x_l \\ x_r \end{bmatrix}^{\mathsf{T}} T^{[1:d]} \begin{bmatrix} x_l \\ x_r \end{bmatrix} + W_{emb} \begin{bmatrix} x_l \\ x_r \end{bmatrix} \right) \tag{3.20}$$

Tensor defined as $T^{[1:d]} \in \mathbb{R}^{2dx2dxd}$ and $[x_l;x_r]^{\mathsf{T}} \in \mathbb{R}^{1x2d}$, where x_l, x_r, h_p are $\in \mathbb{R}^{dx1}$ dimensional word vectors. $T^{[i]}$ is a slice of a tensor $\in \mathbb{R}^{dxd}$. This process is done recursively as defined in the Tree-RNN and φ represents the same tanh function. The primary motivation to use RNTN is to satisfy the direct association between input vectors. The quality of the parent vector is calculated by reconstruction layer as follows:

$$[x_l\prime;x_r\prime] = W_{rec}x_p + b_{rec} \tag{3.21}$$

Here $W_{rec} \in \mathbb{R}^{dx2d}$, $b_{rec} \in \mathbb{R}^{dx1}$. The aim is minimizing the reconstruction error, which is defined as follows:

$$\begin{aligned} E_{EMB} &= L\left([x_l;x_r],\ [x_l\prime;x_r\prime]\right) \\ &= argmin\| \ [x_l;x_r] - [x_l\prime;x_r\prime] \ \|^2 \end{aligned} \tag{3.22}$$

At every internal the parent word vector x_p scored with positive or negative by using softmax function for binary classification task. The phrases are merged recursively to predict sentiment class of the root vector as follows:

$$\begin{aligned} y^{x_p} &= \phi\ (W_{label}x_p) \\ y^{x_p} &= softmax\ (W_{label}x_p) \\ &= \frac{exp(W_{label}x_p)}{\sum_{i=1}^{c} exp(W_{label}\ x_{p_i})} \end{aligned} \tag{3.23}$$

y^{x_p} is polarity prediction, $W_{label} \in \mathbb{R}^{cxd}$ among c sentiment classes. Here $y^{x_p} = softmax\ (W_{label}x_p)$ can be interpreted as a conditional probability $y^{x_p} = p(c|[x_l;x_r])$ and $\sum_{i=1}^{c} y^{x_{pi}} = 1$ and for a given target (true) label t^{x_p} of the parent vector, the total cross-

entropy loss is defined as follows:

$$\begin{aligned} E_{CE} &= -\sum_{i=1}^{c} E(x_{pi}, t^{x_{pi}}, \theta) \\ &= -\sum_{i=1}^{c} t^{x_{pi}} \log(y^{x_{pi}}) \end{aligned} \tag{3.24}$$

The reconstruction error of the compositional embeddings is E_{EMB} and the cross-entropy error of the sentiment labels E_{CE} as follows for a given pair (x^s, t^{x_s}) from the dataset define the total objective function as following:

$$L_{total} = -\frac{1}{N_{out}} \sum_{i=1}^{N_{out}} E(x_i, t^{x_i}, \theta) + \frac{\lambda}{2} \| \theta \|^2 \tag{3.25}$$

Learning happens by using backpropagation through structure (BPTS) with updating the learnable parameters of the model $\theta = (T, W_{emb} W_{rec}, W_{label})$. The comprehensive details of the recursive learning process has been discussed in the next chapter.

3.5 Experiments

3.5.1 Experimental Setup

The reviews are divided into train/dev/test set. The detailed information for each dataset is given at Table 3.6. The hyper-parameters of the model were set following up from previous studies. Socher *et al.* pointed the RNTN model achieved promising results for English when the dimension of the word embeddings was set between 25 and 35. We choosed 30 as a dimension of the word embeddings, which is also used as a dimension of the suffix embeddings [2]. Similarly, the recommended batch size was between 20 and 30. We used the train and dev set to set batch size, learning rate. We observed that the performance of the model was decreasing for larger batch size; hence we used 20 as batch size. 0.01 is used as a learning rate, and AraGrad was used as an optimizer with 0.001 weight decay regularization. The model was trained over 100 epoch. The predictions compared with the baseline methods that were described in the following section. The best accuracy results were obtained over cross validation of dev set.

3.5.2 Baselines

For the sake of fair comparison, MS-TR constructed over datasets that were classified before both by ML and LB methods. The conventional ML algorithms, including NB, SVM, and Max-Ent were used as a baseline methods to compare performances. Feature representation methods for each baseline models are as follows:

Table 3.6 Root and inner node counts of different level annotated MS-TR

Annotation Level	Datasets	Train / Inner Node Count	Dev / Inner Node Count	Test / Inner Node Count
Morph-Level	Books	978 / 98,182	141 / 13,507	280 / 27,252
	DVD	978 / 89,490	141 / 13,644	280 / 27,950
	Electronics	978 / 111,588	141 / 15,208	280 / 32,126
	Kitchen	978 / 95,064	141 / 14,501	280 / 28,376
	Movie	8,000 / 405,964	1,001 / 50,717	1,657 / 84,469
Stem-Level	Books	978 / 86,580	141 / 9,672	280 / 21,1819
	DVD	978 / 78,156	141 / 9,783	280 / 21,368
	Electronics	978 / 90,230	141 / 12,607	280 / 24,034
	Kitchen	978 / 78,309	141 / 10,489	280 / 21,629
	Movie	8,000 / 378,768	1,001 / 45,639	1,657 / 77,460
Token-Level	Books	978 / 73,316	141 / 10,489	280 / 19,878
	DVD	978 / 70,570	141 / 9,979	280 / 19,882
	Electronics	978 / 79,434	141 / 12,783	280 / 23,516
	Kitchen	978 / 70,631	141 / 9,158	280 / 20,332
	Movie	8,000 / 336,008	1,001 / 43,717	1,657 / 73,617
Review-Level	Books	978 / 76,917	141 / 10,647	280 / 21,562
	DVD	978 / 75,836	141 / 14,120	280 / 20,468
	Electronics	978 / 88,550	141 / 16,454	280 / 22,604
	Kitchen	978 / 78,309	141 / 10,489	280 / 21,629
	Movie	8,000 / 379,074	1,001 / 45,618	1,657 / 80,738

- **NB/SVM BoW:** BoW features combined with cross-lingual machine translation feature set from Turkish to English [91].
- **NB/SVM BA:** SentiTurkNet and polarity lexicon, to use the average of words as a feature, basic approach (BA) [114].
- **NB/SVM BA-Neg:** Basic approach combined with handling negation [114].
- **NB/SVM BA-Booster:** Basic appoach combined with booster words [114].
- **NB/SVM BA-Seed Words:** Basic approach combined with handling negation [114].
- **Max-Ent BoW:** BoW features combined with cross-lingual machine translation feature set from Turkish to English [91].

Aforementioned baseline studies did not report the training and test splits ratio. Hence, in this study, we prefer to split train/dev/test dataset like suggested in [2] for STS dataset.

3.6 Results and Discussion

In terms of the experimental results, RNTNs have sentence level (root) and total node accuracy. For the sake of fair comparison, we report the first experiments results as average of the sentence (root) level accuracy of the train set, since the previous baseline studies reported the train dataset results. Table 3.10 and Table 3.11 presents the binary classification accuracy results of the products reviews, and Table 3.7 presents the binary classification accuracy results of the movie reviews. Although Socher *et al.* [2] mentioned that RNTN performed better for shorter reviews, we observed that compared to the traditional methods RNTN performed better even for longer reviews of Turkish movie dataset. In addition to models comparison, we also investigated the effect of different annotation levels to the accuracy results. As it can be seen in Table 3.10, Table 3.11 and Table 3.7, we observed that there is not much difference between the accuracy rates of RNTN over morph-level annotated MS-TR, stem-level annotated MS-TR, token-level annotated MS-TR and review-level annotated MS-TR. However, as shown in Table 3.12, in terms of the inner node accuracy, i.e. all node accuracy, we observed that RNTN performed better with the token-level annotated MS-TR.

Table 3.7 Performance comparisons for movie reviews dataset

Models	**Accuracy %**
RNTN Morph-Level MS-TR	89.60
RNTN Stem-Level MS-TR	**89.84**
RNTN Token-Level MS-TR	89.71
RNTN Review-Level MS-TR	89.57
NB BA	67.49
SVM BA	67.61
NB BA-Neg	68.34
SVM BA-Neg	68.92
NB BA-Booster	69.18
SVM BA-Neg	69.78
NB BA-Seed Words	74.28
SVM BA-Seed Words	75.52
NB BoW	69.5
NB BoW TR-MT	70.0
SVM BoW	66.0
SVM BoW TR-MT	66.5
MaxEnt BoW	68.2
MaxEnt BoW TR-MT	68.6

Table 3.8 Performance comparisons of books dataset

Dataset	Models	Accuracy %
Books	RNTN Morph-Level MS-TR	82.49
	RNTN Stem-Level MS-TR	81.84
	RNTN Token-Level MS-TR	82.80
	RNTN Review-Level MS-TR	**86.08**
	NB BA	67.49
	SVM BA	67.61
	NB BA-Neg	68.34
	SVM BA - Neg	68.92
	NB BA-Booster	69.18
	SVM BA-Neg	69.78
	NB BA-Seed Words	74.28
	SVM BA-Seed Words	75.52
	NB BoW	72.40
	NB BoW TR-MT	72.90
	SVM BoW	66.60
	SVM BoW TR-MT	66.90
	MaxEnt BoW	68.70
	MaxEnt BoW TR-MT	70.50

Table 3.9 Performance comparisons of electronics dataset

Dataset	Models	Accuracy %
Electronics	RNTN Morph-Level MS-TR	83.61
	RNTN Stem-Level MS-TR	82.65
	RNTN Token-Level MS-TR	81.87
	RNTN Review-Level MS-TR	**86.66**
	NB BA	67.49
	SVM BA	67.61
	NB BA-Neg	68.34
	SVM BA - Neg	68.92
	NB BA-Booster	69.18
	SVM BA-Neg	69.78
	NB BA-Seed Words	74.28
	SVM BA-Seed Words	75.52
	NB BoW	73.00
	NB BoW TR-MT	64.40
	SVM BoW	72.40
	SVM BoW TR-MT	64.40
	MaxEnt BoW	74.00
	MaxEnt BoW TR-MT	66.30

Table 3.10 Performance comparisons of DVD dataset

Dataset	Models	Accuracy %
DVD	RNTN Morph-Level MS-TR	81.04
	RNTN Stem-Level MS-TR	**82.95**
	RNTN Token-Level MS-TR	80.84
	RNTN Review-Level MS-TR	82.42
	NB BA	67.49
	SVM BA	67.61
	NB BA-Neg	68.34
	SVM BA - Neg	68.92
	NB BA-Booster	69.18
	SVM BA-Neg	69.78
	NB BA-Seed Words	74.28
	SVM BA-Seed Words	75.52
	NB BoW	76.00
	NB BoW TR-MT	74.90
	SVM BoW	70.30
	SVM BoW TR-MT	67.60
	MaxEnt BoW	71.80
	MaxEnt BoW TR-MT	72.90

Table 3.11 Performance comparisons of kitchen appliances dataset

Dataset	Models	Accuracy %
Kitchen	RNTN Morph-Level MS-TR	81.68
	RNTN Stem-Level MS-TR	80.73
	RNTN Token-Level MS-TR	**81.73**
	RNTN Review-Level MS-TR	79.86
	NB BA	67.49
	SVM BA	67.61
	NB BA-Neg	68.34
	SVM BA - Neg	68.92
	NB BA-Booster	69.18
	SVM BA-Neg	69.78
	NB BA-Seed Words	74.28
	SVM BA-Seed Words	75.52
	NB BoW	75.90
	NB BoW TR-MT	69.60
	SVM BoW	70.00
	SVM BoW TR-MT	67.30
	MaxEnt BoW	72.40
	MaxEnt BoW TR-MT	70.20

Table 3.12 Performance comparisons in term of the total node and sentence-level accuracy

Datasets (Test)	Model Level	All (Node) Accuracy%	Sentence-Level (Root) Accuracy%
Movie	Morph-Level	68.22	89.60
	Stem-Level	68.28	**89.84**
	Token-Level	**68.96**	89.71
	Review-Level	68.48	89.57
Books	Morph-Level	68.86	82.49
	Stem-Level	68.34	81.84
	Token-Level	**73.55**	82.80
	Review-Level	62.14	**86.08**
DVD	Morph-Level	66.92	81.04
	Stem-Level	67.24	**82.95**
	Token-Level	**73.03**	80.84
	Review-Level	59.02	82.42
Electronics	Morph-Level	71.40	83.61
	Stem-Level	70.28	82.65
	Token-Level	**76.24**	81.87
	Review-Level	63.63	**86.66**
Kitchen	Morph-Level	70.83	81.68
	Stem-Level	67.70	80.73
	Token-Level	**75.17**	**81.73**
	Review-Level	68.03	79.86
BOUN 5-class	Review-Level	46.66	43.50

3.7 Summary

This chapter introduces a Morphologically Enriched Sentiment Treebank (MS-TR) for compositional semantics in Turkish. MS-TR was constructed based on the four different annotation levels, including morph level, stem level, token level, and review level using morphological features of the words as a semi-supervised annotation approach. Each annotation level of binary and fine-grained fully-labeled parse trees has been constructed as a novel sentiment treebank. Experiments have been done using different domain datasets with Recursive Neural Tensor Networks (RNTN) and compared to conventional baseline methods, including Naïve Bayes(NB), Maximum Entropy (ME), and Support Vector Machines (SVM). The effect of using labeled stems and suffixes in MS-TR has also been investigated for each dataset. According to the experimental results, RNTN has outperformed conventional baseline methods for each MS-TR annotation level. It has been shown that our semi-supervised distant annotation approach can be practically used to construct a fully-labeled sentiment treebank without the need for human labor while keeping sentiment information of words to construct structured input for Tree-RNNs. This study can be improved using enhanced tree-based deep learning architectures, including

constituency or dependency parse trees. Particularly, the fine-grained train dataset for Turkish can be expanded for future studies.

4
ATTENTIVE COMPOSITION MECHANISMS AND MEMORY BLOCKS OVER RECURSIVE STRUCTURES

"What is the optimal way for compositional representation of sentences?" is still an open question in the NLP field. As we discussed in section 2.4.1, both recursive and recurrent models have a VEG problem when they try to learn over long and deep structures. Although Tree-RNNs are powerful models to learn syntactic and semantic features, they still cannot handle the long-term dependencies, and similar issues could happen like in traditional chain-structured models. LSTM and its variants have been developed to process sequential input bidirectionally or stacked layers [115, 116] to overcome this problem. But these architectures cannot process the tree-structured data similar to the recursive neural networks. Besides, LSTM models also have some disadvantages even they performed well for many tasks, particularly in the field of NLP. They are hard to train, and the same training issues could also happen while backpropagating error over long-term dependencies [99, 117]. Additionally, they cannot fully support batch training and to handle different length sentences, so they need padding operation [118].

This section focuses on extending LSTM to Tree-LSTM, which can work over sentiment treebank while using memory-blocks and attentive mechanism. A novel attentive compositional mechanism has been proposed in binary Tree-LSTMs (ACT-LSMTs) to improve the structural learning ability of the tree-structured LSTMs. ACT-LSTMs have been used to detect the important (more related) part of the long sentences. The motivation is mimicking the human attention mechanism to memorize and learn the more important parts of the given sentences for a downstream task. The aim is to prevent the loss of information, which occurs due to padding operations. In addition, a comprehensive benchmark has also been aimed to compare the performances of advanced chain-structured and tree-structured language models to decide which architecture is better.

Firstly the related work of the attention mechanisms and Tree-LSTM architectures focusing on SA applications have been summarized. Then ACT-LSTM model have been

introduced, which is proposed as an attentive composition mechanism in Binary Tree LSTMs for selective sentiment classification. Next, the benchmarks of the ACT-LSTM model for both chain and tree-structured models have been reported. Finally, last section have concluded with results and discussions.

4.1 Preliminaries

Recently, many researchers have focused on the develop unified and advanced tree-based LSTM architectures [99, 119, 120]. These models have been proposed as an extension of the chain-LSTMs to work over parse trees, and they performed better than the shallow chain structured models. However, besides their compositional advantages, data processing could be very time-consuming due to these architectures do not fully support batched learning [118]. Bowman *et al.* have proposed Stack-augmented Parser-Interpreter Neural Network (SPINN) architecture as a hybrid tree-sequence model [118]. SPINN model gives the opportunity to understand the "depth in space" while learning the "depth in time". It supports batch-learning, and thanks to GPUs, we can make the processing time much faster than previous tree-structured models. GPUs give us a parallel processing chance, so we can process more than one sample at the same time. However, we need to pad inputs at least to the maximum sentence length of each batch, which could cause the loss of information. For example, consider the following two sentences that we use as examples of in the minibatch. S_1 =*Mutlaka izlenmesi gereken harika bir film* "It is a great movie that must be watched." and S_2 =*Berbat bir film.* "It's a terrible movie." We can encode these sentences word by using embedding of the each token such as $S_1 = [x_1, x_2, x_3, x_4, x_5, x_6]$ and $S_2 = [x_7, x_8, x_9]$. Padding is a necessity for batch-learning, let us consider that we put "1" for an absent tokens to represent S_2 as long as S_1 such as $S_2 = [x_7, x_8, x_9, 1, 1, 1]$. It can be clearly seen that this process could cause a loss of information when we need padding for longer and longer sentences.

In this section, an attention mechanism has been proposed to model the important tokens of the sentences to prevent the side-affects of padding operation. Our main motivation is to focus on the many related parts of each sentence for sentiment detection while preserving structural learning over Tree-LSTMs.

Attention layers have been inspired by the cognitive attention mechanisms of humans, which gave superior results in many applications [121]. First remarkable applications of the attention mechanism have been addressed for machine translation task [122] to align translated words. The mechanism has the learning ability without the limitation of the fixed-sized representations of the content while learning an important part of the given text.

Various versions of attention mechanisms have been proposed, including self-attention [123, 124], global-local attention [125], soft-hard attention [126] and they achieved good results for many tasks, [127] including classification tasks [128, 129], NLI [123], machine translation (MT) [124, 125], encoding semantic relations [130], speech recognition [131], recommendation systems [132, 133], and as well as computer vision applications such as image captioning [126]. They have also proved their capabilities for representational learning, such as dynamic word embedding models, which can learn depending on the context. As a well-known transformer model, BERT (Bidirectional Encoder Representation from Transformers) model can learn the contextualised meaning of words in the whole sentences using bidirectional self-attention Transformer (Transformer encoder) [134]. For example, the word "objective" in the sentence "My objective is not just to make an objective speech" should have a different word representation. Or the word apple should have different word embedding for the sentence "I love apple, especially when I buy it from Apple store."

We want to adapt these advantages of attention mechanisms for our own task to improve learning ability Tree-LSTM. Before we introduce the ACT-LSTM model, we give the details of the Tree-LSTMs architectures in the next section.

4.2 Tree-LSTMs and Its Variants

Tree-LSTMs can be considered as a generalization of the basic chain-LSTMs (see Equation 2.32). In this section, two well-known variants of Tree-LSTM have been given. The shallow version of the Tree-LSTM can learn the sentence representation using the following equations:

$$\begin{aligned}
ig_t &= \varphi\,(W_{ih}\,x_t + U_{ih}{}^{L}\,hg^{L}{}_{t-1} + U_{ih}{}^{R}\,hg^{R}{}_{t-1} + b_t{}^{i}) \\
fg_t{}^{L} &= \varphi\,(W_{fh}\,x_t + U_{fh}{}^{LL}\,hg^{L}{}_{t-1} + U_{fh}{}^{LR}\,hg^{R}{}_{t-1} + b_t{}^{f}) \\
fg_t{}^{R} &= \varphi\,(W_{fh}\,x_t + U_{fh}{}^{RL}\,hg^{L}{}_{t-1} + U_{fh}{}^{RR}\,hg^{R}{}_{t-1} + b_t{}^{f}) \\
og_t &= \varphi\,(W_{ho}\,x_t + U_{ho}{}^{L}\,hg^{L}{}_{t-1} + U_{ho}{}^{R}\,hg^{R}{}_{t-1} + b_t{}^{o}) \\
ug_t &= \psi\,(W_u\,x_t + U_u{}^{L}\,hg^{L}{}_{t-1} + U_u{}^{R}\,hg^{R}{}_{t-1}\,b_t{}^{u}) \\
mc_t &= ig_t \odot ug_t + fg_t{}^{L} \odot mc^{L}{}_{t-1} + fg_t{}^{R} \odot mc^{R}{}_{t-1} \\
hg_t &= og_t \odot \psi(mc_t)
\end{aligned} \tag{4.1}$$

Here, the hidden right child vector $hg^{R}{}_{t-1}$, hidden left child vector $hg^{L}{}_{t-1}$, and current state input vector x_t are used as an input to calculate current values of the input gate, left forget gate, right forget gate, and output gate to learn the current and hidden state of the Tree- LSTM cell. As described in [119], we first described the child-sum version, which is useful over dependency trees called Dependency-Tree-LSTMs (DT-LSTMs). It

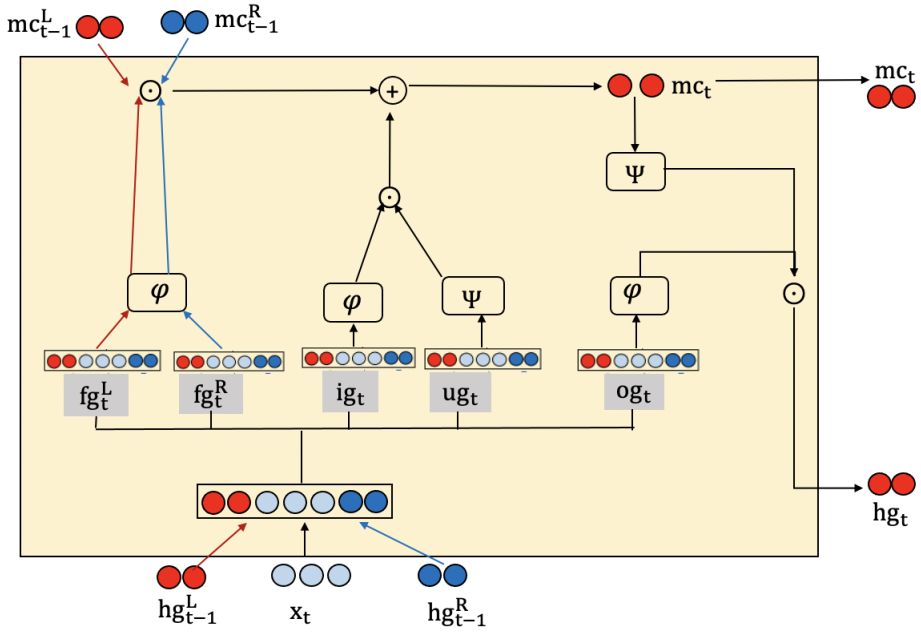

Figure 4.1 Representation of the tree-structured LSTM cell

is operated over child-sums of the tree. Lets Child(k) be a child of node k. Compositional meaning has been learned using the following equations:

$$\begin{aligned}
\widetilde{hg_k} &= \sum_{j \in Childs(k)} h_{kj} \\
ig_k &= \varphi\left(W_{ih}\, x_k + U_{ih}\, \widetilde{hg_k} + b^i\right) \\
fg_{kj} &= \varphi\left(W_{fh}\, x_k + U_{fh}\, h_j + b^f\right) \\
og_k &= \varphi\left(W_{ho}\, x_k + U_{ho}\, \widetilde{hg_k} + b^o\right) \\
ug_k &= \psi\left(W_u\, x_k + U_u\, \widetilde{hg_k} + b^u\right) \\
mc_k &= ig_k \odot ug_k + \sum_{j \in Child(k)} fg_{kj} \odot mc_j \\
hg_k &= og_k \odot \psi(mc_k)
\end{aligned} \tag{4.2}$$

An unordered version of childs, such as dependency relations, are well-performed with the child-sums version. Figure 4.2 represents the child sum Tree-LSTM cell for the childs hg_1 and hg_2. The forget gate number is only two since node k has only two childs. This can be generalised to the k-forget gate for an inner targeted node that has k-child. These forget gates allow to learn selective information for each hidden child.

The second version of Tree-LSTMs is the Constituency-Tree-LSTM (CT-LSTM), that have been proposed for learning compositional meaning over ordered-binary parsed trees, i.e. constituency relations. They are well-performed with N-array variant of Tree-LSTM

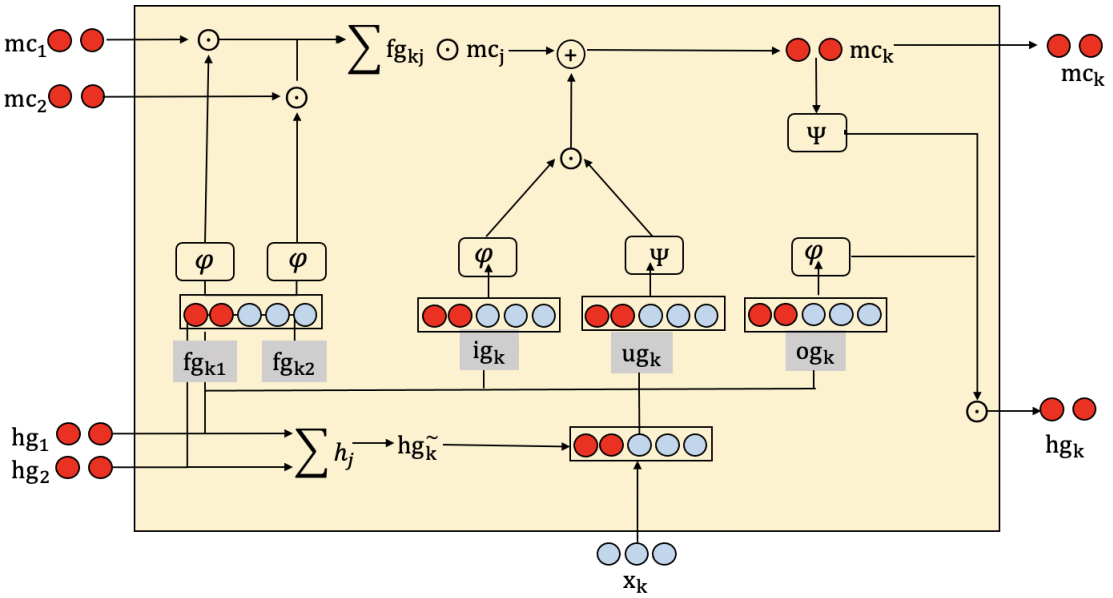

Figure 4.2 Child sum tree topology of LSTM cell at node k for children hg_1 and hg_2

topology and formalised as follows:

$$
\begin{aligned}
ig_k &= \varphi\left(W_{ih}\, x_k + \sum_{l=1}^{N} U_{ih}^{l}\, h_{kl} + b^{i}\right) \\
fg_{kj} &= \varphi\left(W_{fh}\, x_k + \sum_{l=1}^{N} U_{fh}^{jl}\, h_{kl} + b^{f}\right) \\
og_k &= \varphi\left(W_{ho}\, x_k + \sum_{l=1}^{N} U_{ho}^{l}\, h_{kl} + b^{o}\right) \\
ug_k &= \psi\left(W_{u}\, x_k + \sum_{l=1}^{N} U_{u}^{l}\, h_{kl} + b^{u}\right) \\
mc_k &= ig_k \odot ug_k + \sum_{l=1}^{N} fg_{kl} \odot mc_{kl} \\
hg_k &= og_k \odot \psi(mc_k)
\end{aligned}
\tag{4.3}
$$

The leaf nodes are composed to learn the compositional information, which is combined in the hidden layer until to reach the top root of the tree structured sentence. The learning happens by updating the $W_{ih}, U_{ih}, W_{fh}, U_{fh}, W_{ho}, U_{ho}, W_u, U_u, b^i, b^f, b^o, b^u$. Figure 4.3 N-array (2-array) i.e. binary Constituency-Tree-LSTM (CT-LSTM) cell at node k for children hg_1 and hg_2.

Besides their structural processing ability, the trees mentioned above do not have the attention mechanism to capture the more informative words for a specific task. Each child of

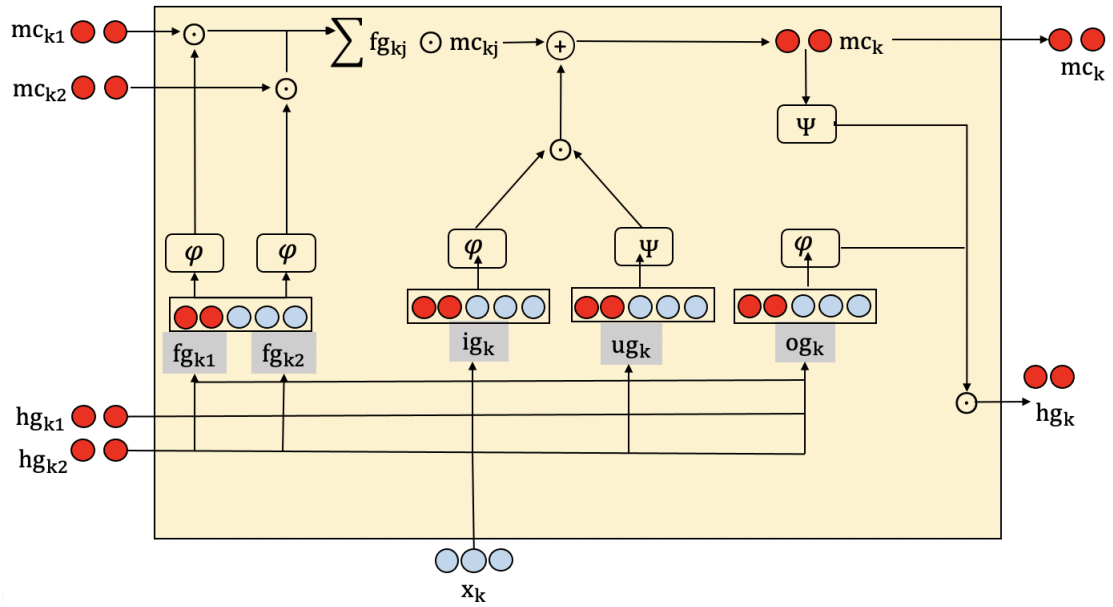

Figure 4.3 N-array (2-array) tree topology of LSTM cell at node k for children hg_1 and hg_2

the tree structure has equal contributions to the compositional meaning of the sentence. In terms of Turkish sentiment analysis, although Recursive Neural Tensor Networks (RNTN) over MS-TR performed well compared to conventional baseline methods, including Naïve Bayes(NB), Maximum Entropy (ME), and Support Vector Machines (SVM), the results are still not well enough and could be improved using enhanced tree-based deep learning architectures, including constituency or dependency parse trees (see Figure 4.4). Particularly, the fine-grained train dataset for Turkish as one of the low-resources language need to be expanded [90]. Hence, we propose an adaptive composition mechanisms in binary Tree-LSTM (ACT-LSTM) as a novel recursive deep architecture model for attentive sentiment distributions. The proposed model extends LSTM to Tree-LSTM, and combines both attention and memory mechanism over recursive tree structures, which learn latent structural information while learning more important part of the sentences.

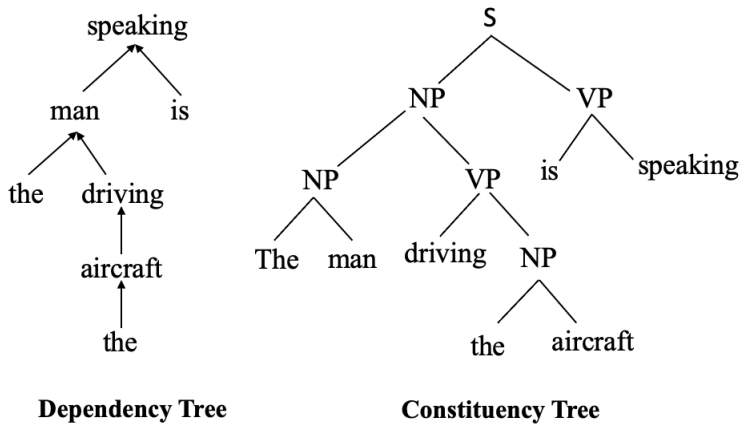

Figure 4.4 Dependency tree and constituency tree of the same sentence

4.3 ACT-LSTMs: Adaptive Composition Mechanisms in Binary Tree-LSTMs for Attentive Sentiment Distributions

4.3.1 Attentive Sentiment Distributions

In this section, we proposed an adaptive tree network combined with an attentive composition mechanism to better understand and strengthen compositional sentiment predictions. For a clear understanding, we first give the basics of the attention mechanism for a shallow chain structured network, as illustrated in Figure 4.5.

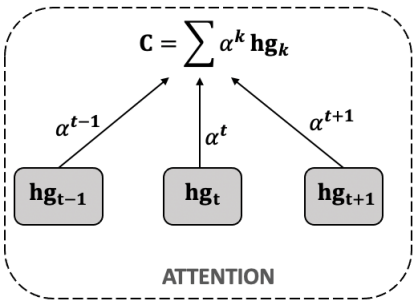

Figure 4.5 Sequential attention mechanism for three time steps of the FFNN

The main idea of the attention mechanism is a weighted summation of the content [122, 124]. The average of the weighted summation of the features proposed for learning a more important part of the context window based on the specified task. Although the attention mechanisms have breakthrough success for encoding-decoding tasks in NMT by the transformers [124], they also contribute to the remarkable performance achievements for various NLP tasks of many networks that have recurrence or memory cells such as RNNs or LSTMs [135].

We can generalize Figure 4.5 using the sequence of d-dimensional word vectors. Let $x^{(s)} = (x_1, x_2, \ldots, x_n)$ represents the tokens of full sentence for $\forall\ x_i \in \mathbb{R}^d\ , 1 \leq i \leq n$.

Attention score [124] for each token is calculated as follows:

$$\begin{aligned} a^k &= \psi(W_a\, hg_k + W_c\, x_c^{(s)}) \\ &= tanh(W_a\, hg_k + W_c\, x_c^{(s)}) \end{aligned} \tag{4.4}$$

where W_a *and* $W_c \in \mathbb{R}^{dxd}$ learnable vector that represents the score of the importance ratio of the given token ψ is the non-linearity scaling function such as tanh. $x_c^{(s)}$ represents the context vector for a given sentence, which is found by chain or tree-topology language model. Dot-product score a_k is used to find the attention weight $\alpha_k \in [0,1]$ is calculated as follows:

$$\alpha_k = \frac{exp(W^T a^k)}{\sum_{k=1}^{n} exp(W^T a^k)} \tag{4.5}$$

where $W^T \in \mathbb{R}^{1xd}$ is the learnable parameter matrix and the weighted representation of the inner states of the $x^{(s)} = (x_1, x_2, \ldots, x_n)$ would be:

$$h_{att} = \sum_{k=1}^{n} \alpha_k hg_k \tag{4.6}$$

Here $h_{att} \in \mathbb{R}^{1xd}$ is transformed by new attentive hidden context representation $\widetilde{h_{att}}$ as follows:

$$\begin{aligned} \widetilde{h_{att}} &= \psi(W_{rep}\, h_{att} + b_{rep}) \\ &= tanh(W_{rep}\, h_{att} + b_{rep}) \end{aligned} \tag{4.7}$$

Recently, there have been some other attention studies that aim to work with tree-topology. Zhou *et al.* tried to encode sentence pairs by embedding attentive layer into Child-Sum-Tree LSTM, and Child-Sum Tree-GRU [136].

4.3.2 Proposed Model

The proposed model unifies the structures and semantics of the sentence using adaptive attentive compositions.

We define the attention mechanism over Binary-Tree-LSTM architecture to detect more important phrases in a sentence. Since finding the optimum compositional function that can learn all the meanings and non-linearity of the dataset is a challenging task [137], we proposed an adaptive composition layer that operates multiple attention functions. The aim is to operate adaptive tree-based attention mechanisms to infer the polarity of the combined words similar to human attention mechanisms. The attention layer operates different attention score functions, namely additive function, general attention, and scaled dot-product attention and concatenation function to find the weighted parent node representation. The attention layer of the ACT-LSTM model randomly selects from those attention score functions, which are defined in Equation 4.14 to define the importance ratio of each child according to the parent vector.

Let hg_L and hg_R be a right and left child of the inner parse tree of the sentence and xp_k be a parent vector of them at node k. We concat left and right child as a hidden key matrix, $M^H = [hg_L; hg_R] \in \mathbb{R}^{dx2}$. Similar to [124], ACT-LSTMs operates a self-attention mechanism, as we search for classification instead of NMT. Hence query (parent) and value (attentive hidden representation) matrices have been chosen equals to key matrix such as $M^H = M^P = M^V$.

In this section, we define the first attentive function operates a scaled-dot product attention score. It is calculated using K_c, Q_p, V_a matrices as follows:

$$K_c = W_c \, M_H \tag{4.8}$$

$$Q_p = W_p \, M_H \tag{4.9}$$

$$V_a = W_v \, M_H \tag{4.10}$$

where the learnable parameter matrices W_c and $W_p \in \mathbb{R}^{dxd}$ and K_c and $Q_p \in \mathbb{R}^{dx2}$.

$$a_1{}^k = \frac{{Q_p}^T K_c}{\sqrt{d}} \tag{4.11}$$

here $a_1{}^k \in \mathbb{R}^{2x2}$ represents the attention score. It is scaled to the probability of the importance of each child of node k as follows:

$$\begin{aligned} \alpha_1{}^k &= \varphi(a_1{}^k) \\ &= softmax(\frac{{Q_p}^T K_c}{\sqrt{d}}) \end{aligned} \tag{4.12}$$

The attention weight matrix $\alpha_1{}^k \in \mathbb{R}^{2x2}$ and values matrix $V_a \in \mathbb{R}^{dx2}$ have been used to generate new attentive hidden representation of the parent value as follows:

$$\widetilde{h_1}^k = \alpha_1{}^k \, V_a{}^k \tag{4.13}$$

where $V_a = W_V M_H \in \mathbb{R}^{2xd}$ and $\widetilde{h_1}^k \in \mathbb{R}^{2xd}$. Here each row of the $\widetilde{h_1}^k$ is an attentive representation of the each children.

The other scaling functions are similar to the aforementioned one, but they use different score functions, which are defined as follows:

$$a_i{}^k = \begin{cases} 1, & \text{scaled dot, } \frac{{Q_p}^T K_c}{\sqrt{d}} \\ 2, & \text{general, } {K_c}^T (W_a Q_p + b_a) \\ 3, & \text{additive, } W_a(K_c + Q_p + b_a) \\ 4, & \text{concatenation, } W_a([K_c; Q_p], b_a) \end{cases} \tag{4.14}$$

here $a_i{}^k$ scaled to the probability of the importance of each child via softmax.

The attention layer can be seen from Figure 4.7, which operates with different scaling functions as they are illustrated in Figure 4.6.

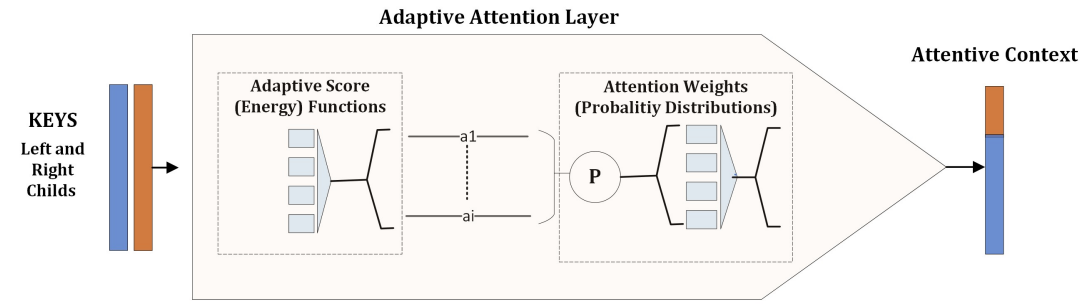

Figure 4.6 An adaptive attention layer of the ACT-LSTM

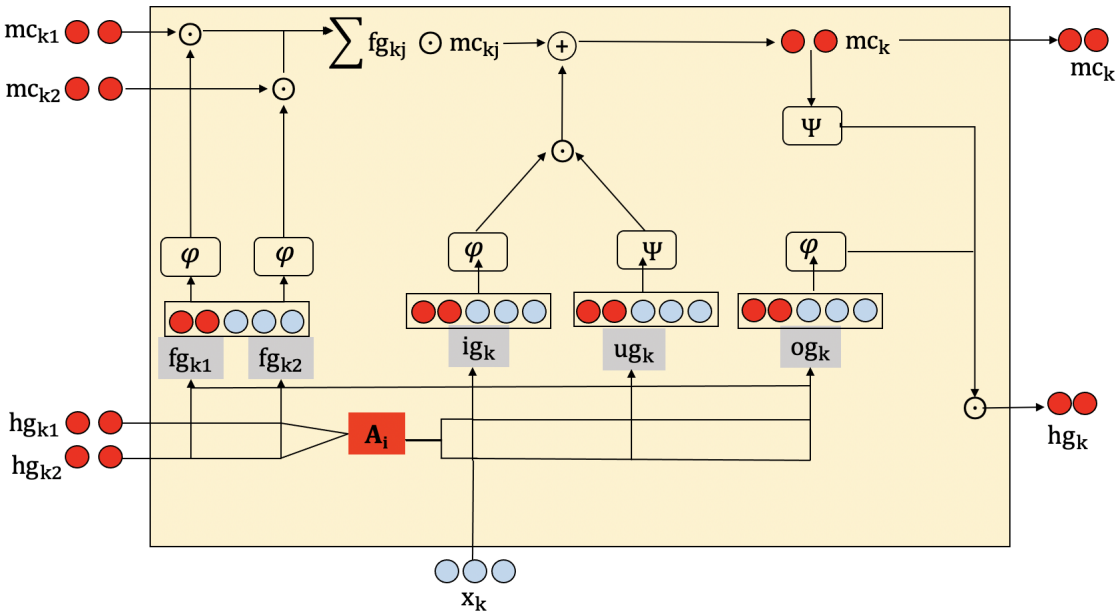

Figure 4.7 An embedded attentive composition mechanism of ACT-LSTM

Equation 4.15 represents the attentive compositional vector of the kth inner of the parsed tree:

$$\begin{aligned} c^{x_{pk}} &= \psi(\varphi(a_i{}^k)V_a{}^k) \\ &= \psi(\alpha_i{}^k V_a{}^k) \\ &= \psi(\tilde{h}_i{}^k) \end{aligned} \tag{4.15}$$

where ψ is the non-linearity function such as tanh and $\alpha_1{}^k, \alpha_2{}^k, ..., \alpha_i{}^k$ are the attention scores. $\varphi(a_i{}^k)$ is the likelihood score of attentive combination of the right and left child vectors which is calculated using adaptive attention scores as given in Equation 4.15 and Equation 4.14. Figure 4.3.3 illustrates the proposed attention mechanism over a binary parse tree.

4.3.3 Training in ACT-LSTMs

For an inner node k, the sentiment prediction of inner node hg_k is calculated as follows:

$$\begin{aligned} y^{[x_k]} &= softmax\ (W_{label}\ hg_k + b_{label}) \\ &= \frac{exp(W_{label}\ hg_k + b_{label})}{\sum_{i=1}^{c} exp(W_{label}\ hg_k^i + b_{label})} \end{aligned} \tag{4.16}$$

Here $y^{[x_k]}$ is the predicted sentiment for an internal hidden node of hg_k. At the same time it can be considered as a conditional probability defined as $y^{[x_k]} = p(c|[x_k])$ and $\sum_{i=1}^{c} y^{[x_k]} = 1$. $W_{label} \in \mathbb{R}^{cxd}$, and c is the number of sentiment class. For a given target (true) label $t^{[x_k]}$ of the hidden node vector, the total cross entropy loss is defined as follows:

$$\begin{aligned} E_{CE} &= -\sum_{i=1}^{c} E([x_k], t^{[x_k]}, \theta) \\ &= -\sum_{i=1}^{c} t^{[x_k]^i} \log(y^{[x_k]^i}) \end{aligned} \tag{4.17}$$

For a given pair (x^s, t^{x_s}) from the dataset, the total cost function is defined by as follows:

$$L_{total} = \frac{1}{N_{out}} \sum_{i=1}^{N_{out}} E(x_i, t^{x_i}, \theta) + \frac{\lambda}{2} \| \theta \|_2^2 \tag{4.18}$$

Here the error $E(x_i, t^{x_i}, \theta)$ is the function for each pair (x^s, t^{x^s}) for a given sentence from the train dataset and it is defined as follows:

$$\begin{aligned} E(x^s, t^{x^s}, \theta) &= \sum_{i=1}^{Totalnodes} E([x_i], t^{[x_i]}, \theta) \\ &= \frac{-1}{Totalnodes} \sum_{i=1}^{Totalnodes} t^{[x_i]} \log(y^{[x_i]}) + \frac{\lambda}{2} \| \theta \|_2^2 \end{aligned} \tag{4.19}$$

$E(x_i, t^{x_i}, \theta)$ is defined as a total error, which comes from the total nodes of each training pair of the dataset that contains the sentiment polarity.

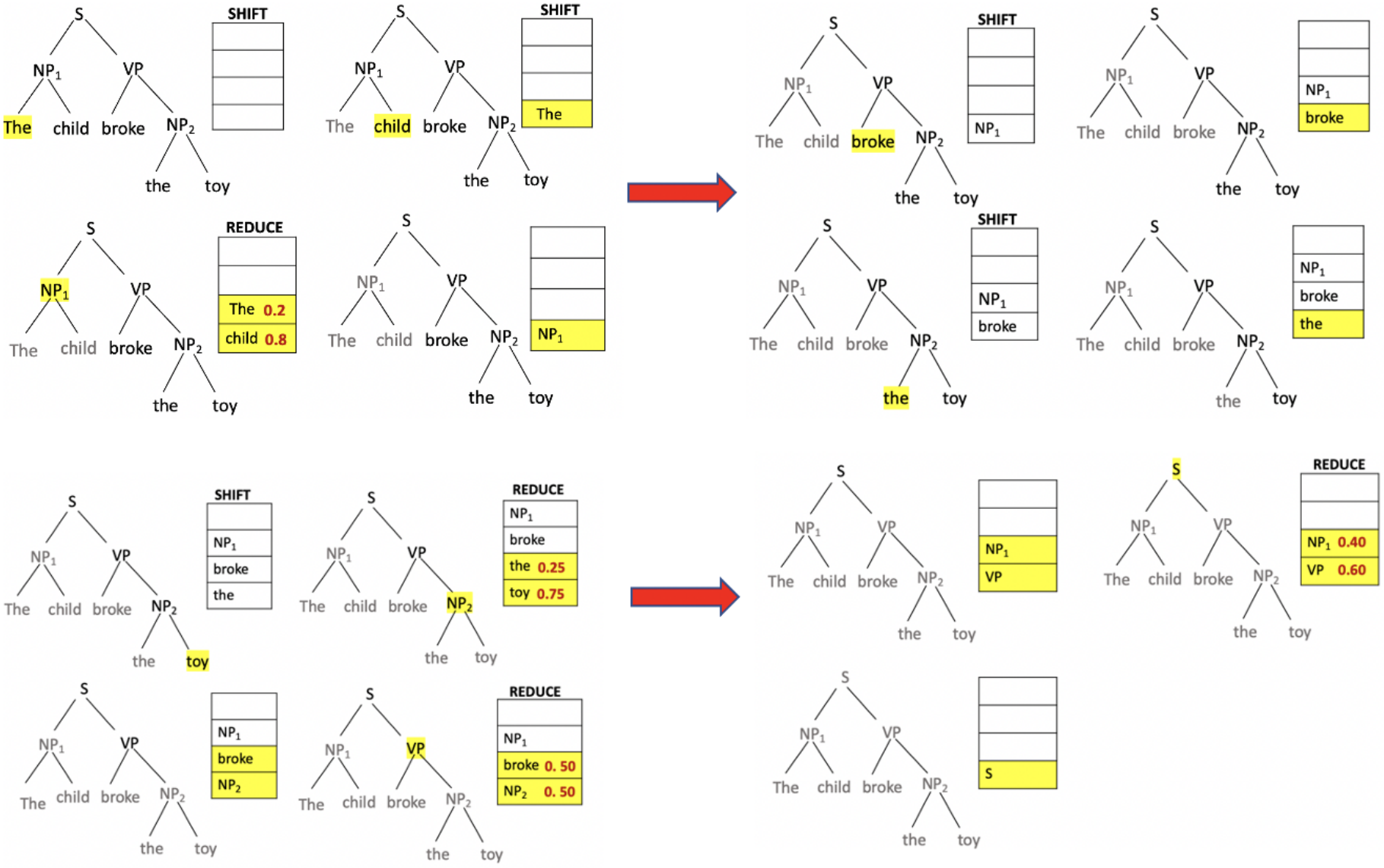

Figure 4.8 Illustration of the attention mechanism over binary tree

4.4 Experiments

4.4.1 Experimental Setup

We have used the same parameters of Recursive Neural Tensor Network (RNTN) for the sake of comparison which is done in [90]. The reviews are divided into train/dev/test set (see Table 3.2 and Table 3.6). The hyper-parameters of the model were set following up from the previous chapter. We chose 30 as a dimension of the word embeddings. Since the performance of the model was decreasing for larger batch size, we used 20 as batch size. 0.01 is used as a learning rate, and AraGrad was used as an optimizer with 0.001 weight decay regularization. The model was trained over 100 epoch. The predictions compared with the chain structured LSTM, tree-structured LSTM, and RNTN were given in the following section. The best accuracy results were obtained over cross-validation of the dev set.

The hyper-parameters of the proposed model have been given in Table 4.1. The maximum sentence length has been selected as 50 based on the observations of the n-grams histograms of the classes (see Figure 4.10). 1003 of BOUN Treebank sentences have been used for classification. The distribution of the five class has been given in Table 4.2.

Table 4.1 Parameter setting

Parameter	Value
$x^{(i)} = (x_1, x_2, \ldots, x_{t-1}, x_t)$	max length = 50
embedding dimension	300
nhidden	150
epoch	100
batch size	32
optimizer	adam
learning rate	2e-4

Table 4.2 Fine-grained dataset

Label	Size
Very Positive	188
Positive	215
Notr	230
Negative	207
Very Negative	163

4.4.2 Baselines

In addition to comparison with conventional ML methods, we propose to compare our model performance with other advanced deep learning models. To this end, we have chosen the best models of the chain-structured models from [138]. We represented them as

CNN-1 to CNN-5 and LSTM-1 to LSTM-5. Moreover, we compared the chain-structured models with Tree-structured LSTM models and also RNTN models, which operate over different-level annotated parsed trees. CNNs and LSTMs models also combined with various segmented features. The details of the models combined with various feature representation methods are as follows:

- **CNN-1 and LSTM-1:** Lemma and suffixes of the words have been used. Suffixes considered as a token similar to the morph-level annotated MS-TR over RNTN-1 [138].
- **CNN-2 and LSTM-2:** Only extracted stems of the words have been used similar to RNTN-2.
- **CNN-3 and LSTM-3:** Only word-tokens have been used similar to the RNTN-3.
- **CNN-4 and LSTM-4:** Hybrid: Words are analysed by [110], and if morphological analysis cannot be done, the word is parsed to its characters [138].
- **CNN-5 and LSTM-5:** The BPE-5k method has been used as a sub-word modelling method [138].
- **RNTN-1:** Tree-structured morph-level annotated MS-TR [90].
- **RNTN-2:** Tree-structured stem-level annotated MS-TR [90].
- **RNTN-3:** Tree-structured token-level annotated MS-TR [90].
- **RNTN-4:** Tree-structured review-level annotated MS-TR [90].
- **ACT-LSTM**: An attentive binary Tree-LSTM model over review-level annotated MS-TR.

In this study, we prefer to split train/dev/test dataset as given in Table 3.6 like suggested in [2] for Stanford Sentiment Treebank (SST) dataset.

4.5 Results and Discussion

Table 4.3, Table 4.4, and Figure 4.9 report the accuracy results for advanced chain-structured models with different segmentation methods, tree-structured models with different annotation levels, advanced tree-structured LSTM models and ACT-LSTM. We have used review-level annotated MS-TR since we found that it gave the best results with token-level annotated MS-TR compared to the stem-level and morph-level annotated MS-TR. According to the experimental results, we observed that Tree-LSTM performed better

compared to the chain-structured LSTM and tree-structured RNTN. RNTN is also better than chain-LSTMs combined with various segmentation methods. Hence we can say that Tree-structured models performed better than the chain-structured models. In terms of the attention layer, even ACT-LSTM performed better than the Tree-LSTM. We cannot observe much more difference between their performances. In addition, CNN combined with word tokenization performed best for the movie dataset. Since the movie dataset is much bigger than the product reviews dataset (see Table 3.2), we can say that CNN with the word token feature could perform better for huge datasets.

Table 4.3 Accuracy results for binary classification datasets

Models	**Book**	**Electr.**	**DVD**	**Kitchen**	**Movie**
CNN Lemma+Suffix	81.43	77.86	76.43	75.00	90.61
CNN Lemma	76.43	73.57	74.29	73.57	88.92
CNN Word Token	77.14	73.57	75.00	77.14	**91.08**
CNN Hybrid	75.71	80.71	75.00	77.14	89.01
CNN BPE-5k	75.71	75.00	75.00	80.71	90.61
LSTM Lemma+Suffix	77.86	77.14	79.29	65.71	90.33
LSTM Lemma	80.00	75.00	77.86	75.00	89.20
LSTM Word Token	77.86	80.71	76.43	75.00	90.80
LSTM Hybrid	75.00	80.00	73.57	74.29	89.48
LSTM BPE-5k	75.71	79.29	75.71	75.00	89.86
LSTM	82.34	72.51	75.11	75.24	83.54
Tree-LSTM	88.34	85.32	84.94	81.97	88.12
ACT-LSTM	**89.49**	**86.77**	**84.99**	**86.08**	89.77
RNTN Morph-Level	82.49	83.61	81.04	81.68	89.60
RNTN Stem Level	81.04	82.65	82.95	80.73	89.84
RNTN Token Level	82.80	81.87	80.84	81.73	89.71
RNTN Review Level	86.08	86.66	82.42	79.86	89.57
NB BoW	72.40	73.00	76.00	75.90	69.50
NB BoW TR-MT	72.90	64.40	74.90	69.60	70.00
SVM BoW	66.60	72.40	70.30	70.00	66.00
SVM BoW TR-MT	66.90	64.40	67.60	67.30	66.00
MaxEnt BoW	68.70	74.00	71.80	72.40	68.20
MaxEnt BoW TR-MT	70.50	66.30	72.90	70.20	68.60

Table 4.4 Average of test accuracy for fine-grained classification dataset

Models	Mean Acc. %
ACT-LSTM	**51.01**
Tree-LSTM	47.31
LSTM	41.56
RNTN	46.66
RNN	34.94

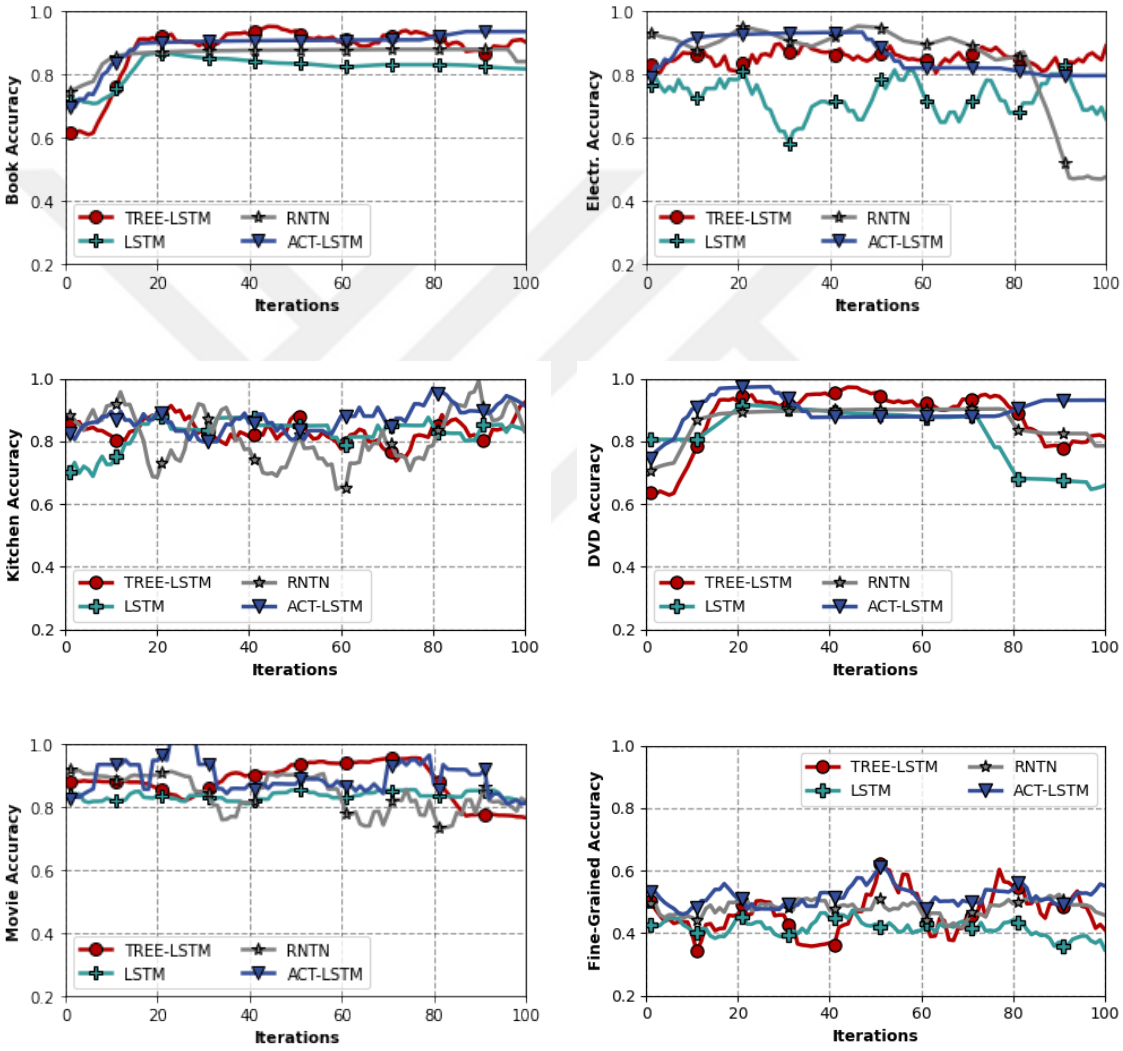

Figure 4.9 Performance comparisons for ACT-LSTM, LSTM, Tree-LSTM and RNTN models

In addition to the comprehensive benchmark, a qualitative analysis has been done to detect the effect of sentence length on the accuracy. Since there is not much more performance difference on binary classification we have chosen to study on a fine-grained dataset. According to the experimental results we observed that ACT-LSTM performed

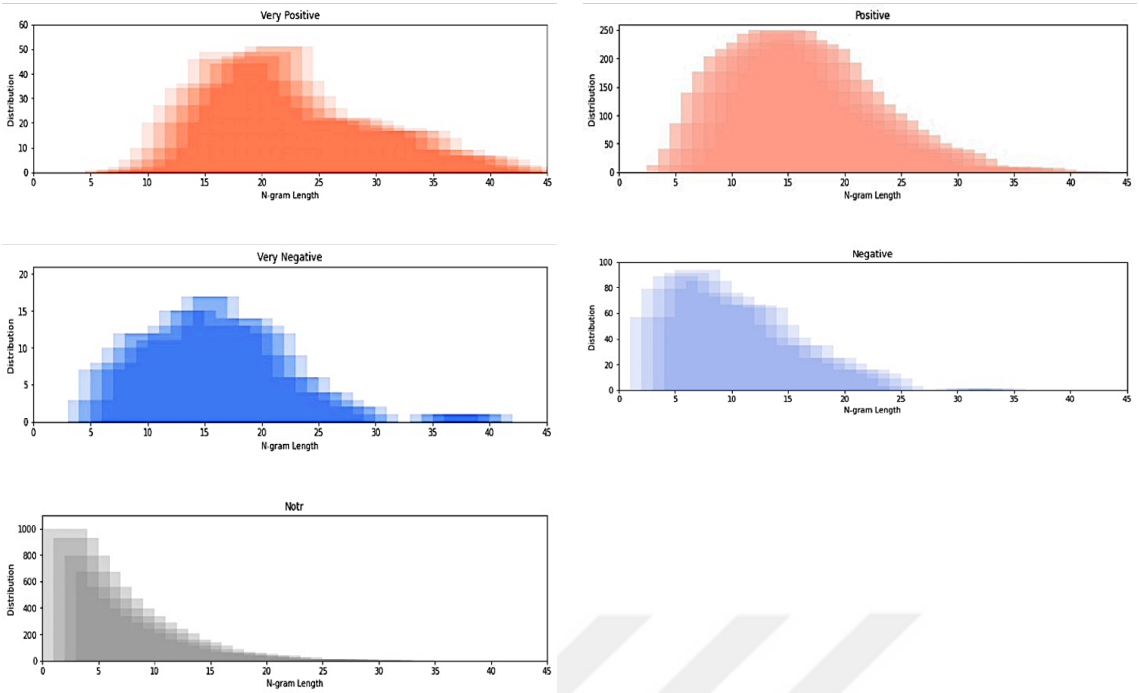

Figure 4.10 Normalized histogram of the annotated n-grams in fine-grained MS-TR

approximately %5 better performance improvement compared to RNTN.

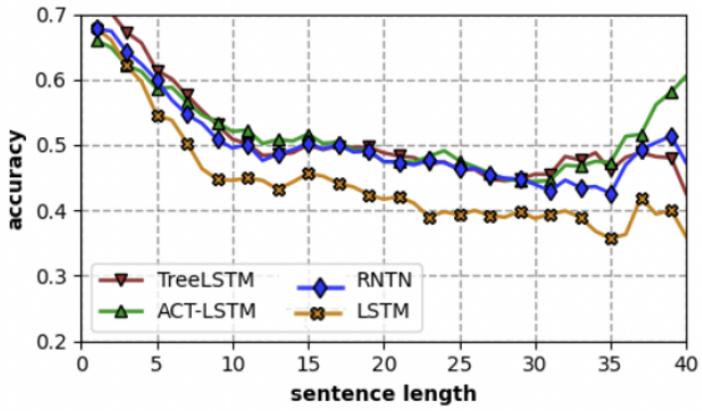

Figure 4.11 Effect of sentence length on the accuracy

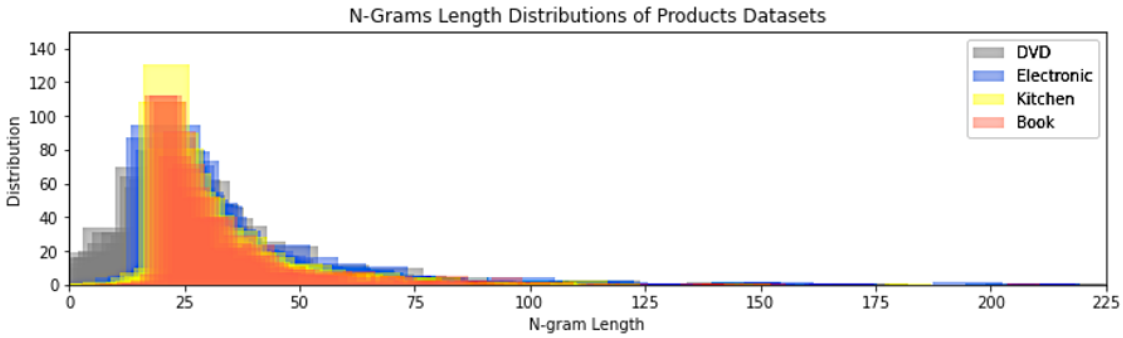

Figure 4.12 N-grams length distributions of products datasets

Figure 4.10 represents the normalized histogram of the annotated n-grams in fine-grained MS-TR and Figure 4.12 represents the n-grams distributions of products dataset. As we can see from Figure 4.11, Tree-LSTM, ACT-LSTM, RNTN performed similar for 10-grams and 35-grams. It has been observed that, ACT-LSTM performed better for longer n-grams. Additional details and the average and maximum N-grams lengths of reviews are given in Table 3.1 and Table 3.2.

4.6 Summary

This section proposed an advanced tree-structured LSTM model for unifying structural learning that supports batch learning to get competitive results for Turkish SA. Although LSTM has many variants to process sequential input bidirectionally or stacked layers, these architectures cannot process the tree-structured data similar to the recursive neural networks. Therefore, we focused on extending LSTM to Tree-LSTM, which can work over a sentiment treebank while using memory blocks and an attentive mechanism. To this end, we proposed a novel attentive compositional mechanism in binary Tree LSTMs (ACT-LSMTs) to improve the structural learning ability of the tree-structured LSTMs.

ACT-LSTMs have been used to detect the critical (more related) part of long sentences. Our primary motivation is mimicking the human attention mechanism to memorize and learn the more important parts of the given sentences for a downstream task, and preventing the loss of information, which occurs due to padding operations. Experiments have been done over a Morphologically Enriched Turkish Sentiment Treebank (MS-TR), and a comprehensive benchmark has been done over performances of advanced chain-structured and tree-structured, and convolutional deep learning models to detect which architecture is better.

According to the experimental results, ACT-LSTM performed better in terms of fine-grained Sentiment Analysis (SA). We observed that Tree-LSTMs performed better com-

pared to chain-structured LSTMs and RNTN. RNTN also performed better than chain-LSTMs combined with various segmentation methods. Hence we can say that tree-structured models performed better than the chain-structured models. In terms of the attention layer, even ACT-LSTM performed better than the Tree-LSTM, we could not observe much more difference between their performances in terms of binary classification. However, it is demonstrated that tree-structured models are better than chain-structured models for binary and fine-grained sentiment classification.

5
METAHEURISTICS FOR TRAINING DEEP SEQUENTIAL RECURSIVE LANGUAGE MODELS

This section focuses on the deep sequential recursive language models, namely Recurrent Neural Networks (RNNs), which have been modelled for learning information from temporal sequences. So far, due to the limitations of the conventional language models, there are many advanced tree-structured, and chain-structured neural language models have been proposed. All these models were actually developed because of the underlying, well-known training problems. As it is well-known, designing an optimum deep recurrent neural network is difficult due to configuration and training issues such as the VEG problem (see section 2.4.1). In this section, it is investigated whether deep RNNs will be as good as previous advanced models if their learning capabilities are improved. To this end, a novel metaheuristic optimisation approach is proposed for training deep RNNs for the sentiment classification task. The approach employs an enhanced ternary Bees Algorithm (BA-3+) which maintains low time-complexity for large dataset classification problems by considering only three individual solutions iteratively. BA-3+ combines the collaborative search of three bees, performing local learning with exploitative search, SGD learning with singular value decomposition (SVD), and global learning with explorative search. Thus, the algorithm utilises the greedy selection strategy of the local search operators of the basic Bees Algorithm to improve solutions, the stabilisation strategy of SVD to handle the problem of VEG of the decision parameters, and the random global search strategy of the basic Bees Algorithm to achieve faster convergence avoiding getting trapped at local optima. BA-3+ has been used to detect the optimal value of trainable parameters of the proposed deep recurrent learning architecture. The proposed algorithm has been compared for sentiment detection. According to the experimental results, the improved accuracy and convergence results showed that the proposed algorithm performed better compared to traditional SGD and BA-3+ is an efficient algorithm for training deep RNNs for complex classification tasks.

5.1 Preliminaries

Deep RNNs are powerful models which can learn from the large sets of sequential data for various tasks in NLP [139], time series prediction [140], machine translation [141] and image captioning [142]. Deep RNNs have self-looped connected deep layers, which can retain knowledge from the previous time steps and can learn from arbitrarily long time sequences. However, despite their theoretical power, they have well-known computational issues such as training difficulties due to VEG [46], the need for implementation in hardware and memory limitations [56]. Besides, designing a deep learning model to perform a particular task could be very time-consuming as it involves many optimisation steps such as selecting a proper network architecture, finding the optimum hyper-parameters of the selected architecture, and choosing the correct training algorithm for the model.

Training a deep RNN is making it learn higher-level nonlinear features from large amounts of sequential data, which is typically a nonconvex optimisation problem [56]. This problem can be formulated as the minimisation of nonlinear loss functions with multiple local optima and saddle points. From the perspective of optimisation, even convex optimisation problems have many challenges. Additional difficulties therefore arise in training DNNs because of the nonconvex nature of the problem. For example, SGD is a commonly used training algorithm, could easily get trapped at local minima or saddle points, and it cannot guarantee convergence to the global optimum because of the nonlinear transformations in each hidden layer. Moreover, the gradient of nonlinear activation functions cannot be computed backward through the network layers without vanishing or exploding over many training time steps [47, 143], which causes the loss of direction in parameter updating to reach a feasible solution.

So far, researchers have mainly focused on two alternative pathways to deal with long-term dependencies. The first pathway is to devise new network architectures such as LSTMs [44], GRUs [48] and Temporal Restricted Boltzmann Machines (TRBM) [144]. Although these architectures have proved successful in many applications, they are more complex to implement and require long implementation and computation times, in addition to specialised software and powerful hardware. The second pathway is to develop search methods and optimisation algorithms specifically to handle the vanishing and exploding gradient problem. Recently, two popular methods, gradient clipping and gradient scaling, were proposed to avoid the gradient explosion issue. Gradient clipping [46] which employs a shrinking strategy when the gradient becomes too large, is used to avoid remembering only recent training steps. Shrinking has also been employed by second-order optimisation algorithms, but these have been replaced by simple SGD as a fair and practical technique because of the computational cost of Hessian matrices in second-order optimisation [116].

The learning performance of deep learning models does not depend only on improving the training algorithm. The initial design parameters also play a key role in the ability to find global optima without becoming trapped at local stationary points. For example, the initial weights of model can significantly affect training performance and good solutions often cannot be reached with gradient-based training algorithms because of the nonlinearity and "butterfly-effects" of the iterative updating procedure [46]. Generally, design parameters are adjusted manually, and the designer has to evaluate the model performance repeatedly to determine the best objective functions, learning rates, or training algorithm for their task. Besides, even when the optimal model could be designed, additional regularisation strategies such as dropout [145] are required to handle the overfitting problem of a deep model. It is well-known that these procedures are very time-consuming, and new strategies are needed to develop practical solutions.

Numerical methods and exact algorithms cannot handle the nonconvexity of the objective functions of deep RNNs, which are unable to capture curvature information, causing the optimisation process to be trapped at local solutions. Nature-inspired metaheuristic algorithms have been developed to handle nonlinear, multi-constraint and multi-modal optimisation problems. They have proved to be robust and efficient optimisation tools that can avoid the issue of local optima. They can adapt to problem conditions like the nature of the search space (i.e. continuous or discrete), decision parameters, varying constraints and other challenges encountered in the training and designing of RNN models. Previous research into the optimisation of deep learning models has focused on three main areas, namely, hyperparameter optimisation, neural architecture or topology optimisation, and weight optimisation. Cai *et al.* used particle swarm optimisation (PSO) and evolutionary algorithm (EA) (hybrid PSO-EA) as a hybrid algorithm for training RNNs for time series prediction [73]. A recurrent NARX neural network has been trained by a Genetic Algorithm (GA) to improve the state of charge (SOC) of lithium batteries [146]. The NeuroEvolution of Augmenting Topologies (NEAT) approach has been developed based on the GA [147] for the optimisation of neural model architectures. Desell *et al.* have used Ant Colony Optimisation (ACO) to design a deep RNN architecture with five hidden and five recurrent units for predicting flight data [64]. Similarly, Ororbia *et al.* have implemented Evolutionary eXploration of Augmenting Memory Models (EXAMM) and different versions of it such as GRU, LSTM, MGU and UGRNN [148] to evolve RNNs.

However, there have only been limited studies into optimising the architecture of a deep RNN [64] or deep LSTM [76]. For example, the authors of [64] have used ACO to convert fully-connected RNNs into less complex Elman ANNs. These studies have been conducted for specified tasks with the numbers of hidden and recurrent neurons limited to a maximum of five, and new practical approaches are needed to be useful for deeper RNN models [149].

This section proposes using an enhanced ternary Bees Algorithm (BA-3+) to obtain the optimum weights of a deep RNN model for sentiment classification. Existing population-based optimisation algorithms need to operate with large populations and, as a result, are generally slow. The Bees Algorithm [150] is a population-based algorithm that has been successfully employed to solve complicated optimisation problems [151–153]. It is able to find optimum solutions without needing to calculate the gradient of the objective function. The ternary Bees Algorithm (BA-3) first described in [154] is an improvement on other population-based algorithms that employs a population of just three individual solutions. The BA-3+ algorithm presented in this section is an enhanced version of BA-3+ that also uses only three individual solutions, the global-best solution, the worst solution and an in-between solution. BA-3+ combines the exploration power of the basic Bees Algorithm to escape from local optima and the greedy exploitation drive of new local search operators to improve solutions. The new local search operators comprise one for neighbourhood search using SGD and one for search control employing SVD. SGD is a greedy operator for reaching a local optimum quickly. SVD is adopted to stabilise the trainable parameters of the model and overcome the problem of VEG of the selected weights when SGD is applied to derive the in-between solution. The aim is to use the strengths of gradient-based backpropagation training as the most commonly used RNN training method, but without its limitations like local optimum traps and VEG through long time dependencies. As the proposed algorithm uses only three individual bees, it is very fast, being able to find the global optimum within polynomially-bounded computation times [152, 154]. Experiments with the sentiment classification of English and Turkish movie reviews and Twitter tweets show that the ternary BA performs well, providing faster and more accurate results compared to previous studies.

We first start with the reviews applications of metaheuristics to the training of deep RNNs. Section 3 presents detailed information about deep RNNs and the difficulties with training them. Section 4 details the proposed algorithm and its local search operators, and describes its configuration for training deep RNNs for sentiment classification. Section 5 provides information about the datasets used, the hyper-parameters of the model, and the experimental results obtained. Section 6 concludes the section.

5.2 An improved Bees Algorithm (BA-3+) for Training Deep Recurrent Networks

In this section, a population-based search algorithm for training deep RNNs is presented. The learnable parameters $\theta = (W_{xh}, W_{hh}, W_{hy}, b_h, b_y)$ are the same as in SGD, which is defined as a candidate solution in BA-3+, and try to minimise the binary cross-entropy loss function $L(y,t)$ for each pair of the sequential input $(x_1, x_2, \ldots, x_{t-1}, x_t)$, the desired-

targeted output t, and the predicted value y.

Gradient-based learning algorithms are particularly sensitive to the initial value of the weights and noise variance of the dataset in nonconvex optimisation. Hence, the difficulty of the training deep RNN model depends on not only keeping the information through long-term time but also initial values of the parameters. Most initialisation methods are generally based on the random initialisation [155] or researchers choose to initiate the weights as an identity matrix or close to the identity conventionally [156]. Therefore, finding optimum initial parameters for a specified model and exploring which parameters should be updated and learned are still remains an open difficult optimisation task, due to the lack of the exact knowledge about the which properties of these parameters are kept or learned, under which conditions [56].

As mentioned above, this work uses an enhanced Ternary Bees Algorithm (BA-3+) for training deep RNNs. BA-3+ combines exploitative local search with explorative global search [154]. Improvements to the training of deep RNN models with BA-3+ have been made in three key areas: finding promising candidate solutions and initialising the model with good initial weights and biases, improving local search strategies to enhance good solutions by neighbourhood search, particularly to handle the VEG problem, and performing exploration to find new potential solutions with global search.

5.2.1 Bees Algorithm

The Bees Algorithm has been implemented by Pham *et al.*, which models the clever food-foraging behaviours of the honey bees[150]. Honey bees have the population-based searching abilities to exploit good food sources. Every honey bee can extend itself in multiple directions and distances which can be more than 10 km from hive to explore new flower patches which are abundant in pollen or nectar. The honey bees have a good memory to remember the quality of the visited food source in terms of location of the flower patch, its distance to the hive or its food abundance like sugar content. They also have good communication mechanisms to share their experiences with other follower bees when they return to the hive. They do "waggle dance" or wag-tail dance [157] to communicate about their exploration journey and tells other bees "where" the visited good patches is [158]. The follower bees observe this dance, and they are recruited to go to the specified patches by the distance and the direction information of the selected patches. This fascinating dance language enables the communication between honey bees and starts the recruitment process of the bees' behaviours. It becomes guidance of the exploration, local search of potentially profitable flower patches for the unemployed bees. A larger number of bees follow the experiences of the dancer bees and the colony of honey bees may maximise the benefits of the recruitment process.

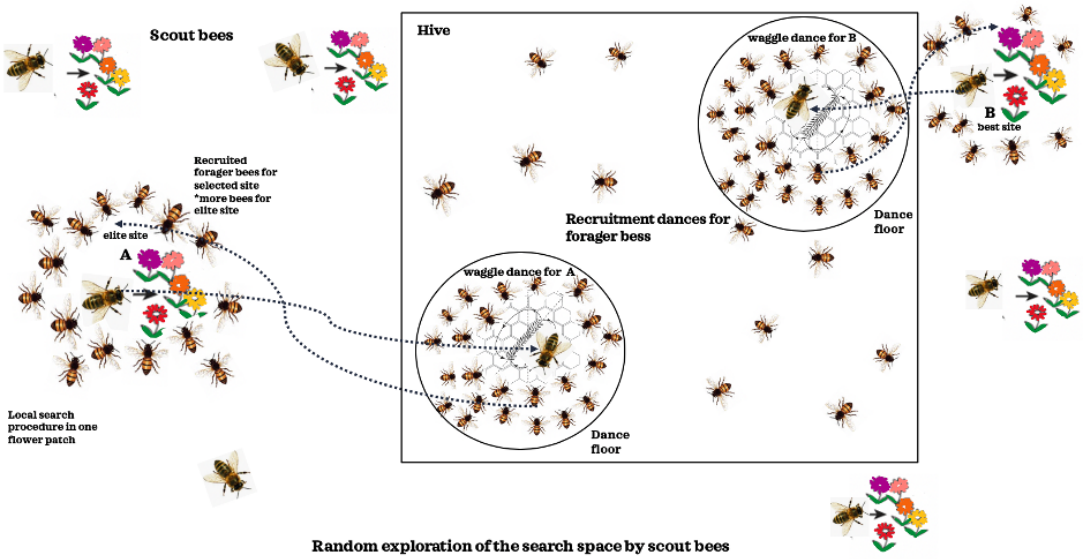

Figure 5.1 Foraging behaviour of honey-bees for local search and global search phase

5.2.2 Representation of Bees for Deep RNN Model

The "Bee" in the proposed method represents a sequential deep RNN model, which is modelled for binary sentiment classification task. As it can be seen in Figure 5.3, every Bee (Sequential model) instance has the input layer, hidden deep RNN layers, and the output layer. The proposed model has the learnable parameters $\theta = (W_{xh}, W_{hh}, W_{hy}, b_h, b_y)$, and aim to classify the sequential input data $(x_1, x_2, \ldots, x_{t-1}, x_t)$ to its targeted class t. Based on the training procedure of the RNN model, each "Bee Model" has its own forward propagation action to calculate the initial solutions, and local search procedure, gradient descent training with SVD, and global search actions to find the optimal parameters $\theta = (W_{xh}, W_{hh}, W_{hy}, b_h, b_y)$ via the binary cross-entropy loss function (fitness function) $L_{BCE}(f(x^{(i)}, \theta), y^{(i)})$ as defined Section 2.3.3.1.

BA-3+ does not require a large population, which is a drawback with other population-based methods. BA-3+ employs only three individual bees for each training time. Each iteration begins with these three initial solutions as a forward pass of the model and continues with specified search strategies including exploitative local search, SGD stabilised by SVD, and explorative global search.

As with the basic Bees Algorithm [150], the initial candidate solutions are sorted. The maximum fitness value is selected as the best RNN bee for the local exploitative search. The worst fitness value (third bee) is selected for global search to avoid getting trapped at local optima, and the remaining RNN, i.e. the middle RNN bee is selected for stochastic gradient-descent learning with the stabilisation strategy of SVD operator to update weights and biases without VEG.

Algorithm 9: The pseudo-code of the basic Bees Algorithm (BA) for continuous domains

Input: n: ScoutBee, m: SelectedBee, e: EliteBee,
nsp: SelectedSitesBee, nep:SelectedEliteSitesBee, neighbourhoodSize: ngh
Function `BA` (*nScout,m,e,nsp,nep, ngh*) **:**

```
population ← InitialSolutions(ScoutBee)
while stopping criterion not met do
    Evaluate fitness of the population
    Sort population according to fitness function
    Select m best solution for local search
    // Generate local solutions with neighbourhood
     search
    for each Bee ∈ e do
        for each Bee ∈ nep do
            localBee ← Bee + random(−ngh, ngh)
            localBee ← Fitness(localBee)
            if localBee better than Fitness(Bee) then
                // Update Bee
                Bee = localBee
    for each bee ∈ m − e do
        for each bee ∈ nsp do
            localBee ← Bee + random(−ngh, ngh)
            localBee ← Fitness(localBee)
            if localBee better than Fitness(Bee) then
                // Update Bee
                Bee = localBee
    // Assign remaining bees for global search
    for each Bee ∈ n − m do
        globalSolutions ← GenerateRandomSolutions(Bee)
    Evaluate fitness of the new population
    Select Best Bee from the new population
return BestBee
```

Fig.5.2 represents the flowchart of the proposed algorithm. $(x^{(train)}, t^{(train)})$ is the training sample from the dataset, that $x^{(i)} = (x_1, x_2, \ldots, x_{t-1}, x_t)$ is defined as an n-dimensional sequential input and $t^{(i)}) \in \{0, 1\}$ is its targeted sentiment class. Three initial solutions are calculated with the initial trainable parameters θ (see Table 5.2) and sorted according to the loss function. The elite (best) RNN bee performs the local search operator, the middle RNN bee performs SGD with SVD operator, and the third RNN bee performs global search. The optimisation continues with a new population of bees until the stopping criteria met; in other words until the loss value is converged to zero.

Table 5.1 Parameters of the deep RNN model trained by using BA-3+

Parameters	Information
$x^{(i)} = (x_1, x_2, \ldots, x_{t-1}, x_t)$	i^{th} temporal sequential input from the input training set
$t^{(i)}$	i^{th} targeted class of the output training set
$y^{(i)}$	i^{th} predicted class of the $x^{(i)}$
W_{xh}	Weight matrix from input layer to hidden layer
W_{hh}	Weight matrix from hidden layer to hidden layer
W_{hy}	Weight matrix from hidden layer to output layer
b_h, b_y	Biases for the hidden layer and the output layer
nScout	Number of scout bees for initialisation
ngh	Neighbourhood size for the local search
nhidden	Hidden layer size
η	Learning rate for SGD

Table 5.2 Learnable (trainable) parameters

Parameters	Dimension
W_{xh}	$\mathbb{R}^{number\ of\ hidden\ layers\ \times\ dimension\ of\ each\ word}$
W_{hh}	$\mathbb{R}^{number\ of\ hidden\ layers\ \times\ number\ of\ hidden\ layers}$
W_{hy}	$\mathbb{R}^{\ \lvert V\rvert \times\ number\ of\ hidden\ layers}$
b_h	$\mathbb{R}^{\ number\ of\ hidden\ layers}$
b_y	$\mathbb{R}^{\ number\ of\ output\ layers}$
$\lvert V\rvert$:	number of words in the vocabulary

5.2.3 Local Search Operator

The local search procedure in the basic Bees Algorithm includes improving the promising solution within the "flower patch" (see Fig.5.1), which represents the neighbourhood of the selected solution parameters. In the following pseudo-code of algorithm 11, *ngh* represents the initial size of the neighbourhood for local search after waggle dance.

The neighbourhood begins as a large area and it is reduced by using a shrinking method [152] at each iteration, which is defined as $ngh(t+1) = \alpha\ ngh(t)$. Here, α is usually a number between 0 and 1. The neighbourhood matrix is generated with the same dimension of each weight matrix of the learnable parameters $\theta = (W_{xh}, W_{hh},\ W_{hy}\ , b_h, b_y)$, and then is aded to the original weight matrix to obtain the updated weights. The pseudo-code to generate neighbourhood weights is given in algorithm 10. The updated local weights are used for the local search of BA-3+ that can be seen in algorithm 10.

5.2.4 Enhanced Local Search by SGD and Singular Value Decomposition (SVD) Operator

As analysed in Section 2.3.3.1, due to the sharing the same hidden matrix W_{hh} across the deep hidden layers and multiplying it again and again at every time step of the BPTT al-

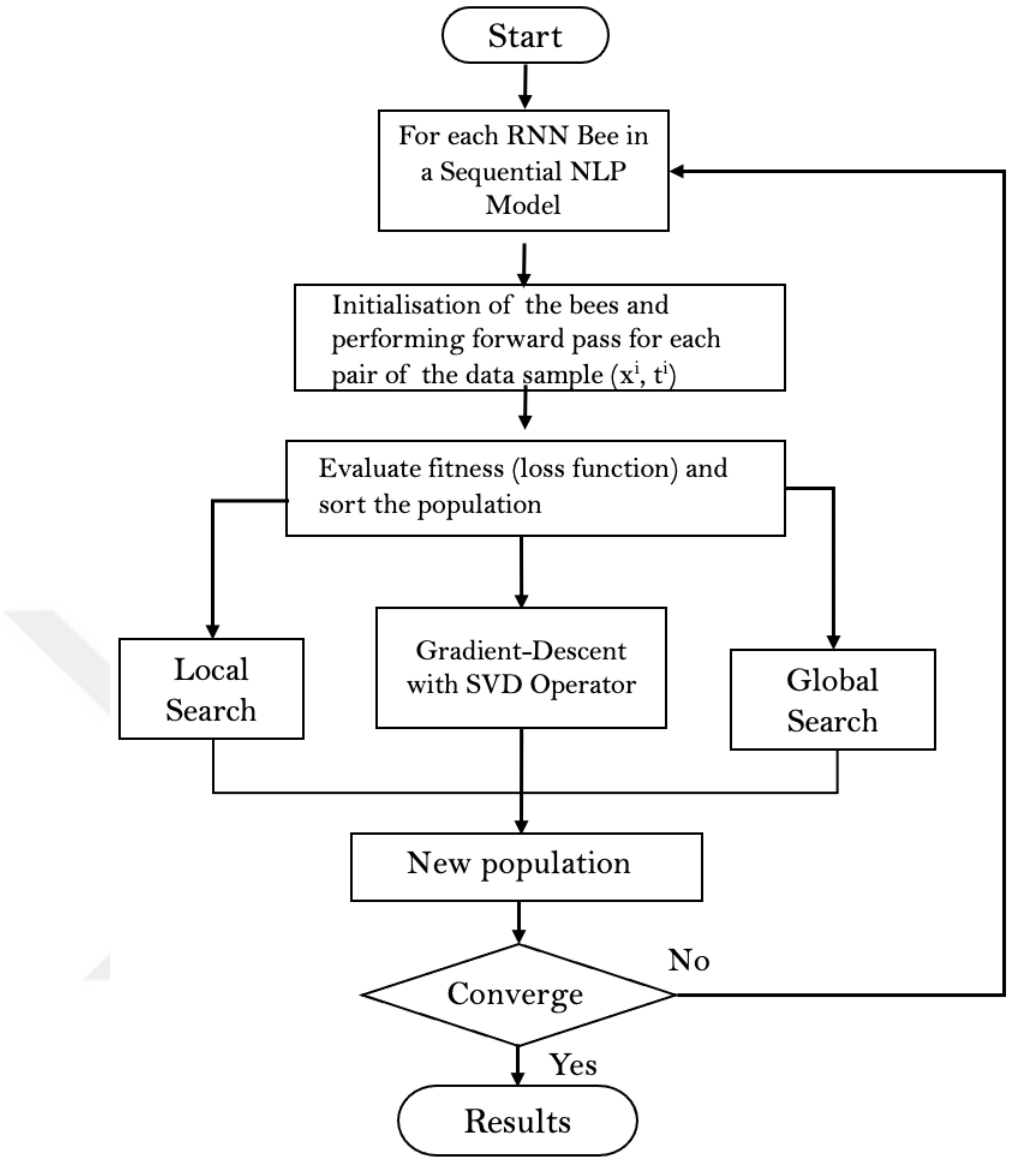

Figure 5.2 Flowchart of the proposed enhanced ternary Bees Algorithm (BA-3+)

gorithm, the eigenvalues of the Jacobian matrix exponentially grow or vanish after t time steps. To handle this issue, it has been proposed to use a singular value decomposition of the hidden layer matrix to stabilise the eigenvalues of the updated matrix in the enhanced local search of BA-3+. As an example, assume that the eigenvalues of the W_{hh} are represented $\lambda_1, \lambda_2, \ldots, \lambda_n$ The singular values of W_{hh} can be founded by using the positive eigenvalues of the matrix $W_{hh}{W_{hh}}^T$, for every $\lambda_i \geq 0 \in \lambda_1, \lambda_2, \ldots, \lambda_n$, $S_i = \sqrt{\lambda_i}$ [159] if W_{hh} is positive semi-definite square matrix. Since the learnable parameters of the RNN can also be rectangular matrices, it is needed to find singular values of an arbitrary matrix A.

It is well-known that every arbitrary real matrix can be represented by the product of three matrices as $A = USV^T$, which is called singular value decomposition (SVD) of matrix

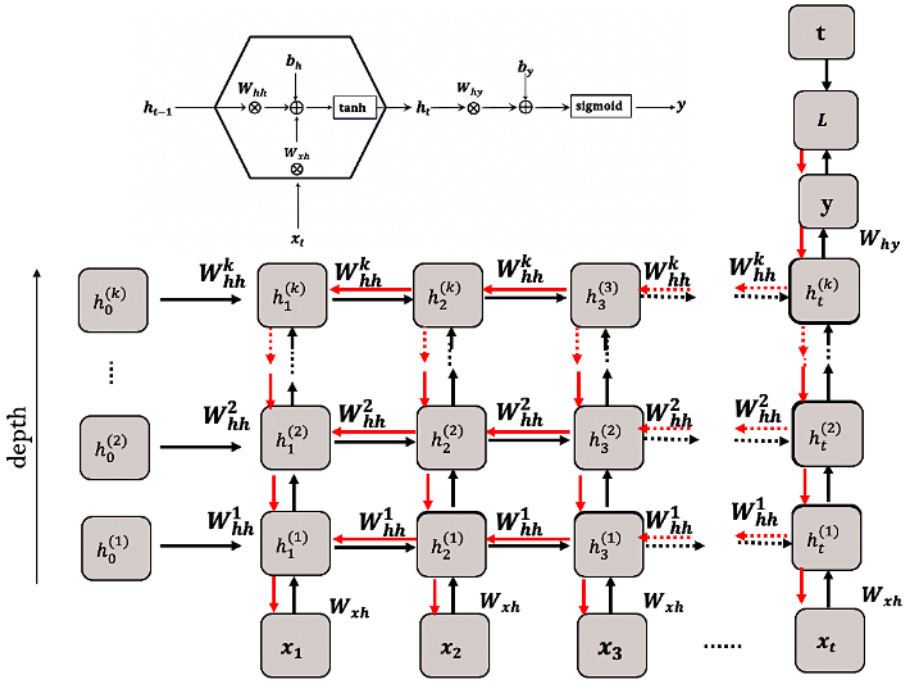

Figure 5.3 A deep RNN architecture representing a bee in the proposed algorithm. Black lines are the forward pass of RNN cell at time t (unfolded version at upper) and red lines representing the error backpropagation through long-term dependencies

Algorithm 10: Generate neighbourhood weights

Input: *weight matrix* : w, ngh
Output: *updated ngh weights*
Function `generateNghWeight` (w, ngh) **:**
  for *each* $w_e \in w$ **do**
    $ngh \leftarrow random\ number \in [-ngh, ngh]$
    $w_{ngh} \leftarrow w_e + ngh$
  return w_{ngh}

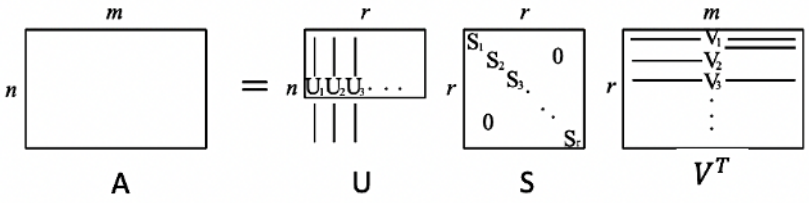

Figure 5.4 Singular value decomposition (SVD) of matrix A

A, which is used to find the singular values [160]. Figure 5.4 represents the SVD of an $n \times m$ dimensional matrix. Here, S is the $r \times r$ dimensional diagonal matrix $S_{n \times n} = diag[S_1, S_2, \ldots, ..., S_n]$ that each S_i represents the singular values of the matrix A, and U and V contain the corresponding singular vectors where U and V are orthogonal matrices with the $n \times r$ and $r \times m$ dimensions, respectively.

Algorithm 11: Local search after waggle dance

Input: *Bee*, *ngh* : *neighbourhood radius*
Output: *Local Bee with Updated Parameters*
Function `LocalSearch(Bee, ngh)` **:**
  for *each* $w \in \theta = (W_{xh}, W_{hh}, W_{hy}, b_h, b_y)$ **do**
    $dim_w \leftarrow dimension(w)$
    $w_{ngh} \leftarrow generateNghWeight(w, ngh)$
    $Bee.w \leftarrow w_{ngh}$
  return *Bee*

Algorithm 12: SGD with SVD

Input: Learning rate:η, *Bee*, *ngh* : *neighbourhood radius*
Output: *Bee with Updated Parameters*
Function `SGDSVD(`η, θ`)` **:**
  for *each* $w \in \theta = (W_{xh}, W_{hh}, W_{hy}, b_h, b_y)$ **do**
    `// Update` θ `by using gradient descent rule`
    $\theta^{t+1} = \theta^t - \eta \nabla_\theta L(f(x_i, \theta), y_i))$
    Calculate SVD for each w :
    U, S, V^T = SVD(w)
    for *each* $s_i \in S$ **do**
      if $s_i \geq (1 + ngh)$ **then**
        $s_i = 1 + ngh$
      else if $s_i \leq 1/(1 + ngh)$ **then**
        $s_i = 1/(1 + ngh)$
    $Bee.w \leftarrow w_{SVD}$
  return *Bee*

After updating each parameter of the $\theta = (W_{xh}, W_{hh}, W_{hy}, b_h, b_y)$ by SGD rule, the SVD operator has used to control the eigenvalues of each parameter. The method aims to keep the singular values of the updated matrix close to 1 for gradient stabilisation. To this end, the SVD decomposition of the updated matrix is performed to find the singular values, and then every singular vector is controlled to be close to the unit vector.

As given in Figure 12., the singular values of the updated weight matrix are restricted to the interval $[1/(1 + ngh), 1 + ngh]$ to avoid updating in the wrong direction. Here, *ngh* is the initial neighbourhood size, which is chosen between (0,1). As a result, W_{hh} can be updated over-time without vanishing or exploding gradients.

5.2.5 Global Search Operator

Besides the enhanced local search procedures, the proposed algorithm also includes a global search operator that combines random sampling chances which is also a good strategy for escaping local optimum points of the solution space. The third bee in a colony

Algorithm 13: An improved ternary Bees Algorithm for training deep RNN model

Input: nScout, learning rate:η, ngh: neighbourhood radius, dataset

Function BA−3+ ($nScout, \eta, ngh, dataset$) **:**

Start
$inputs \leftarrow CreateInputs(dataset)$
$targets \leftarrow labels(y)$
$items \leftarrow$ *convert dataset to list of* `// x=sentences and t=targets`
for *each* $(x^{(i)}, t^{(i)}) \in items$ **do**
Initialize *population with ternary* RNN_{Bee}
while *stopping criterion not met* **do**
Evaluate fitness of the population
y, $Loss \leftarrow$ FORWARD$(RNN_{Bee}, x^{(i)})$
Sort population according to loss values
$local_{bee} \leftarrow$ LocalSearch$(best_{Bee}, ngh)$
Evaluate fitness of $local_{Bee}$
if $local_{Bee}$ *better than* $best_{Bee}$ **then**
`// Update First Bee`
$best_{Bee} = local_{Bee}$
$SGDSVD_{Bee} \leftarrow$ SGDSVD$(second_{Bee}, ngh)$
Evaluate fitness of $SGDSVD_{Bee}$
if $SGDSVD_{Bee}$ *better than* $second_{Bee}$ **then**
`// Update Second Bee`
$second_{Bee} = SGDSVD_{Bee}$
$global_{Bee} \leftarrow$ GlobalSearch$(third_{Bee})$
Evaluate fitness of $global_{Bee}$
if $global_{Bee}$ *better than* $third_{Bee}$ **then**
`// Update Third Bee`
$third_{Bee} = global_{Bee}$
Evaluate fitness of the new population
Sort population according to loss values
Best Model = Best Bee
return *Best Model (Best Bee)*

is used for the random exploration for potential new solutions of the search space. If the updated random weights gave a better solution for the loss function, then the third bee is updated with new global searched weights. This procedure gives the advantage to escape the getting trapped at local optima, that results with converging to the global optimum faster during the training process. Figure 5.2 shows the pseudo-code of the proposed enhanced Ternary Bees Algorithm (BA-3+). Algorithm 13 shows the pseudo-code of the proposed enhanced Ternary Bees Algorithm (BA-3+).

5.3 Experiments

5.3.1 Experimental Setup

The proposed algorithms were implemented using the Tensorflow library with Keras Sequential model in Python on the macOS Catalina on MacBook Pro, 3.1 GHz quad-core Intel Core i5 hardware. The proposed BA-3+ algorithm was run with batch size 1 for each $(x^{(i)} = (x_1, x_2, \ldots, x_{t-1}, x_t), t^{(i)})$ pair of datasets. Each dataset was divided into a training set and a validation set using 5-fold cross-validation. The training was performed with BA-3+ and SGD according to BCE loss value over 100 independent runs, each involving 100 epochs. The BA-3+ training procedure was online learning and happened incrementally over each iteration, which means the learnable parameters were updated after each forward and backward propagation of each training sample [161]. Figure 5.5 represents the flowchart of the proposed classification model.

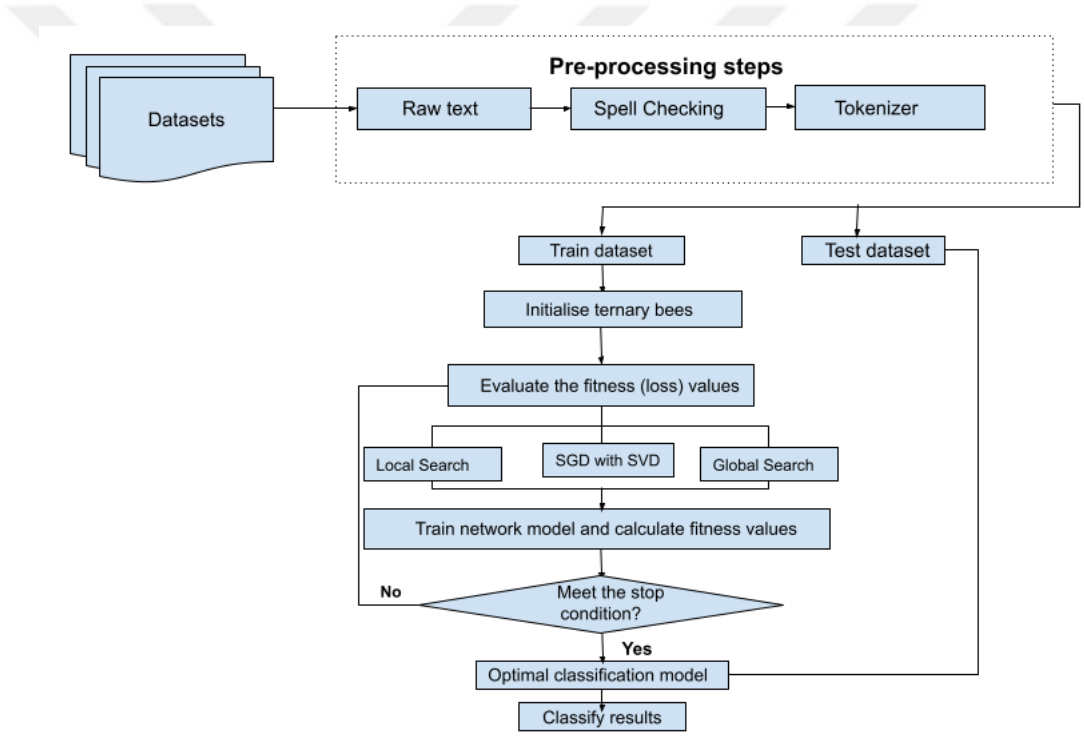

Figure 5.5 Flowchart of the proposed classification model

Table 5.3 shows the parameters of BA-3+. The number of scout bees represents the number of sequential RNN networks. Each training sample of the input data $x^{(i)} = (x_1, x_2, \ldots, x_{t-1}, x_t)$ is padded to the maximum sequence length after preprocessing steps of the given dataset. Widely-used sentiment classification datasets in Turkish and English from 3 different domains (movie reviews, multi-domain product reviews and twitter reviews) were adopted to verify the proposed algorithm. Table 5.4 presents the datasets information in detail.

In all of the experiments, the sequential model was implemented with the same hyper-

Table 5.3 Parameter setting

Parameter	Value
$x^{(i)} = (x_1, x_2, \ldots, x_{t-1}, x_t)$	max length = 200
nScout	3
nEiteSiteBee	1
nSelectedSiteBee	1
ngh	0.5
nhidden	32
epoch	100
independent runs (solution numbers)	100
batch size	1
upper limit of max singular value	$1+ngh$
lower limit of min singular value	$1/(1+ngh)$
η (learning rate)	0.01

Table 5.4 Turkish and English datasets

Dataset	**Size**
TR Books [91]	700 P, 700 N
TR DVD [91]	700 P, 700 N
TR Electronics [91]	700 P, 700 N
TR Kitchen Appliances [91]	700 P, 700 N
Turkish Movie Reviews [91]	5331 P, 5331 N
Turkish Twitter Dataset [162]	12490 P, 12490 N
English IMDB Movie Reviews [163]	12500 P, 12500 N
English Movie Reviews [1]	1000 P, 1000 N
English Yelp dataset [164]	3337 P, 749 N

parameters for the sake of fair comparison. Every sequential model was constructed with the input layer, hidden RNN layers, and the output layer. The embedding layer was used as the input layer, which converts the indexes of the sequential input to the fixed size dense vectors as an input of the model. The vocabulary size of each dataset was used as the input dimension of the embedding layer. The random uniform function was used as weight initialiser for both the embedding layer and hidden layers.

As a critical hyper-parameter, the neighbourhood size (ngh) of the BA-3+ is set to 0.5. The learning rate (η) of the SGD is set to 0.01. The neighbourhood size is used for both the local search and the stabilisation of the largest and the smallest eigenvalue of the updated parameters. The number of hidden layers is set to 32 for each model. Each element of the learnable (trainable) parameter of the $\theta = (W_{xh},\ W_{hh},\ W_{hy},\ b_h,\ b_y)$, the dimension of the weight matrices is changed according to hidden layer numbers, and the corresponding dimensions of the weight matrix can be seen in Table 5.2, in Section 5.2.3.

In this section, F1-score or F1 measure was used as a statistical measure for the analysis of the binary classification problem in addition to the accuracy measure. F1 measure is calculated as follows:

$$F_1 = \frac{2\ Precision\ Recall}{Precision + Recall} \tag{5.1}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{5.2}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{5.3}$$

$$Accuracy = \frac{TP + TN}{(TP + TN + FP + FN)} \tag{5.4}$$

Precision is the total count of true positives divided by the total number of positive results. The recall is the total number of true positive results divided by the number of all samples that should have been classified as positive. F1 measure can also be defined as a harmonic mean of the precision and recall value. The next section reports performance results and benchmarks.

5.4 Results and Discussion

Table 5.7 reports the best and average accuracy, loss and F1 measures obtained by both the BA-3+ and SGD algorithms. The accuracy and loss values of the training and validation datasets are given as percentages. The average values were calculated after 100 independent training runs. Figure 5.6 shows the loss values for each dataset after one independent run which contains 100 training epochs. According to experimental results, the proposed BA-3+ algorithm guarantees fast convergence to the optimum value and the lowest error rate for each dataset.

Figure 5.7 and Figure 5.8 represent the distributions of the loss values for both BA-3+ and SGD over 100 independent runs. Table 5.7 reports the best and average accuracy

and loss values of the training datasets and validation datasets. BA-3+ performs better compared to traditional SGD. Results showed that BA-3+ can be used as an effective learning method for the sentiment classification task. The performances of BA-3+ were better both in terms of the best and average accuracy compared to SGD. Similarly, the best and average values of BA-3+ were lower than for SGD for each dataset.

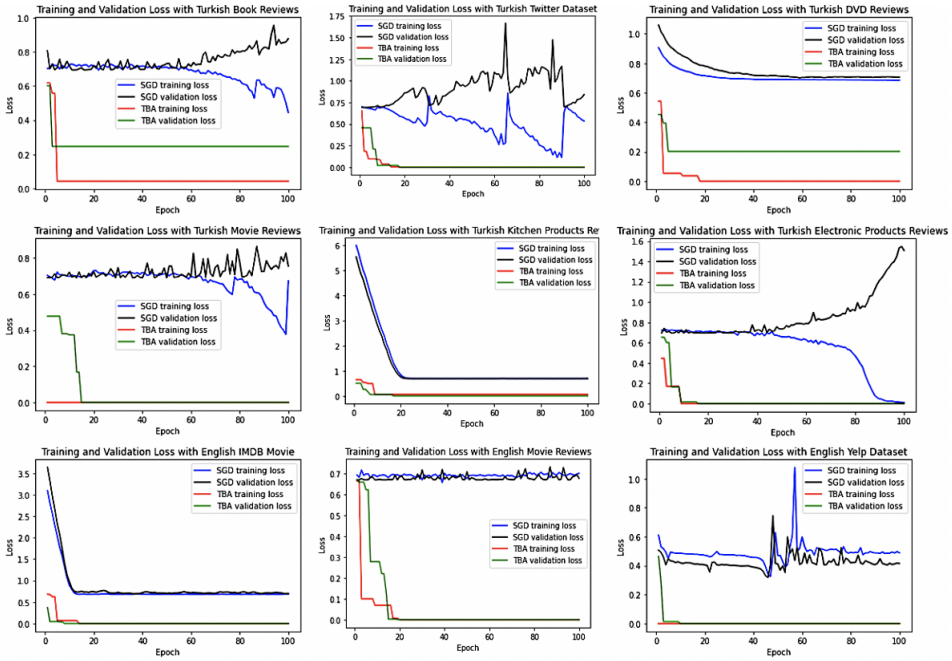

Figure 5.6 The comparison of loss values based on BA-3+ and SGD

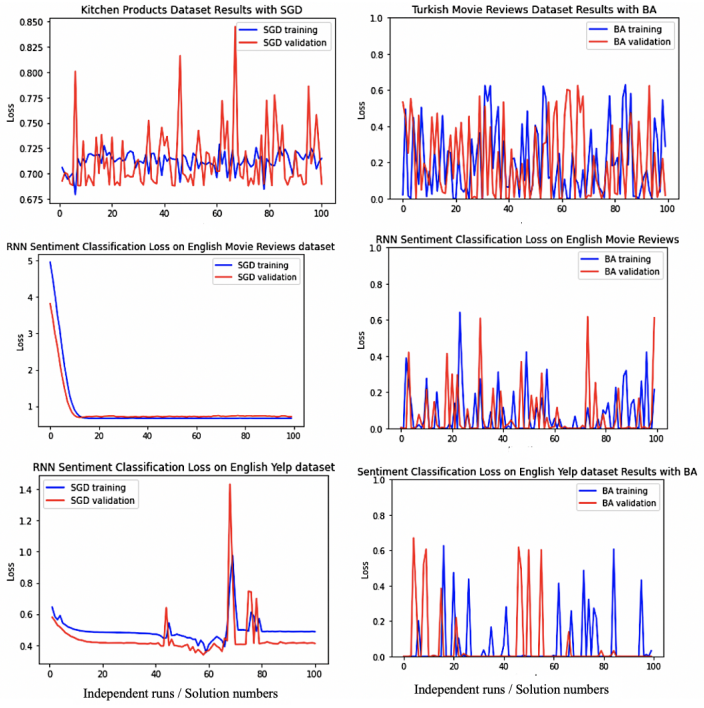

Figure 5.7 Distributions of the loss values of the training and validation dataset for 100 independent runs

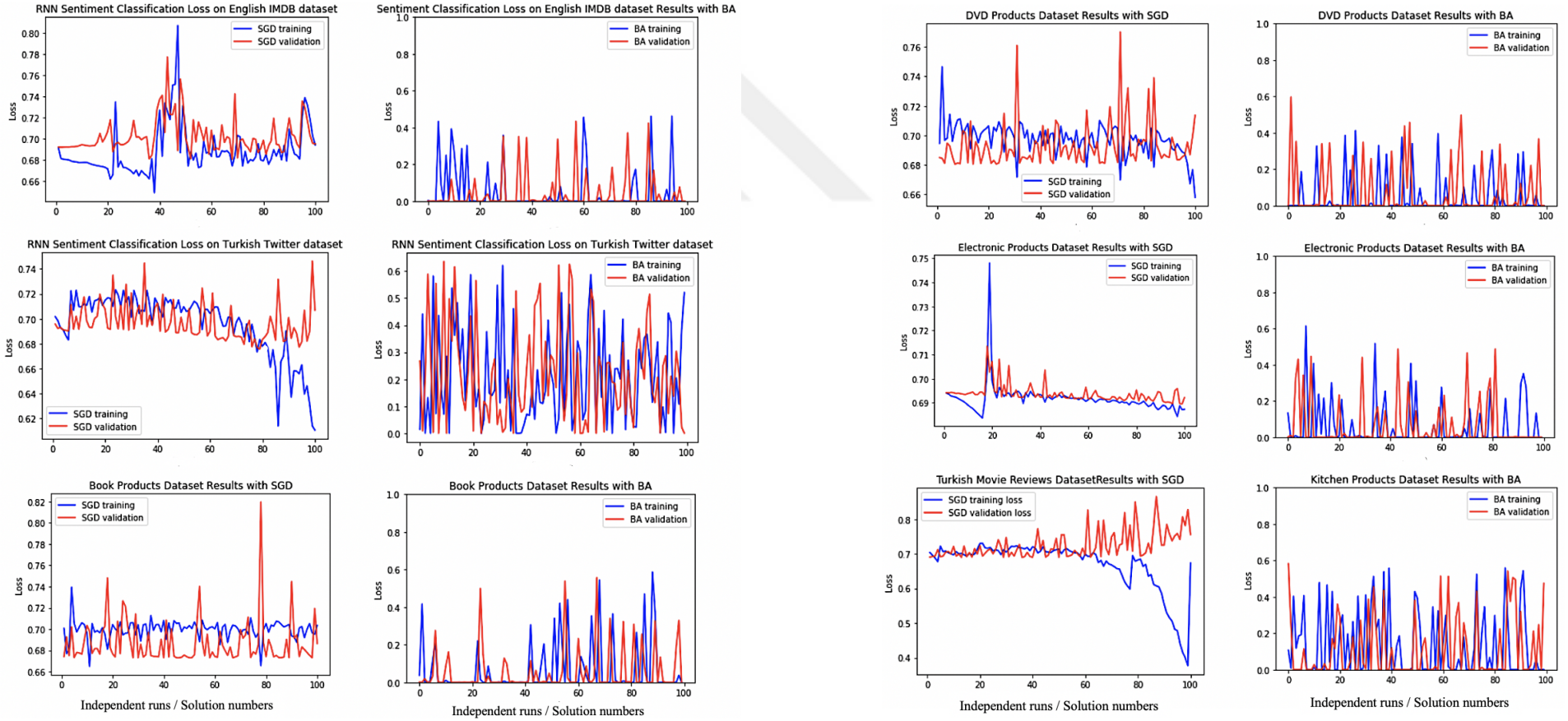

Figure 5.8 Distributions of the loss values of the training and validation dataset for 100 independent runs

Table 5.5 Total training time (sec) of the BA-3+, DE, PSO and SGD algorithms

Datasets	SGD Total Time	BA3+ Total Time	DE Total Time	PSO Total Time
TR Book	237.36	393.139	871.608	544.914
TR DVD	233.92	407.187	860.439	520.700
TR Elect.	233.15	411.297	1317.971	512.482
TR Kitchen	235.73	417.765	855.125	513.929
TR Movie	1021.38	1394.631	3827.576	3969.091
TR Twitter	4334.98	8340.9	16178.52	8978.6
EN IMDB	12100.00	22347.5	31724.76	25674.5
EN Movie	1293.718	2206.8	3361.95	2399.71
EN Yelp	4691.78	12112.8	21157.6	13894.9

Table 5.6 Comparison of BA-3+ performance with DE and PSO and SGD

Datasets	SGD Acc.	BA3+ Acc.	DE Acc.	PSO Acc.
TR Book	0.527	0.801	0.789	0.690
TR DVD	0.519	0.923	0.808	0.678
TR Elect.	0.58	0.915	0.786	0.696
TR Kitchen	0.507	0.81	0.794	0.686
TR Movie	0.58	0.93	0.887	0.759
TR Twitter	0.747	0.914	0.74	0.709
EN IMDB	0.579	0.91	0.778	0.701
EN Movie	0.585	0.896	0.78	0.69
EN Yelp	0.79	0.87	0.71	0.68

BA-3+ has been compared with Differential Evolution (DE) and Particle Swarm Optimization (PSO) algorithms. For the sake of comparison, PSO has been evaluated with three particles as we propose to employ only three scout bees over 100 epochs. However, DE has been employed with 100 generations since it gave too low accuracy when it employees with only three generations. Table 5.5 reports the time consumption of the SGD, BA-3+, DE, and PSO algorithms in second. As can be in Table 5.6, BA-3+ has outperformed the DE and PSO, and its computation time of the BA-3+ is lower from DE and PSO. Although BA-3+ time consumption was longer than standard SGD, the accuracy value and F1 measure have improved for all classification datasets, at least with a 30% - 40% improvement. Furthermore, BA-3+ was more stable and converged faster than the DE and PSO algorithms since it employs only three individual bees as a population. As is expected SGD model has the lowest computation time, but the accuracy result is also lower compared to all other metaheuristic training algorithms.

Table 5.7 Comparison of the results of 100 independent experiments with 100 epochs

		Accuracy Results				Loss Results				F1 Score	
Datasets	Alg.	$Train_{Best}$	Val_{Best}	$Train_{Avg}$	Val_{Avg}	$Train_{Best}$	Val_{Best}	$Train_{Avg}$	Val_{Avg}	$Train_{Avg}$	Val_{Avg}
TR Book	BA-3+	0.99	0.99	0.801	0.882	0.00	0.00	0.0769	0.117	0.81	0.78
	SGD	0.73	0.64	0.527	0.51	0.686	0.688	0.695	0.697	0.68	0.67
TR DVD	BA-3+	0.99	0.99	0.923	0.91	0.00	0.00	0.076	0.089	0.80	0.78
	SGD	0.52	0.39	0.519	0.389	0.686	0.701	0.708	0.748	0.70	0.70
TR Elect.	BA-3+	0.99	0.99	0.915	0.916	0.00	0.00	0.084	0.083	0.83	0.81
	SGD	0.99	0.58	0.615	0.549	0.092	0.684	0.640	0.849	0.75	0.73
TR Kitchen	BA-3+	0.99	0.99	0.81	0.810	0.00	0.00	0.189	0.191	0.81	0.75
	SGD	0.52	0.54	0.507	0.525	0.692	0.689	0.640	0.813	0.70	0.67
EN IMDB	BA-3+	0.99	0.99	0.911	0.90	0.00	0.00	0.032	0.006	0.77	0.75
	SGD	0.58	0.47	0.579	0.469	0.68	0.693	0.81	0.870	0.71	0.70
TR Twitter	BA-3+	0.99	0.99	0.914	0.830	0.00	0.00	0.017	0.029	0.76	0.73
	SGD	0.99	0.55	0.747	0.456	0.112	0.691	0.51	0.909	0.59	0.57
TR Movie	BA-3+	0.99	0.99	0.93	0.855	0.00	0.00	0.00	0.05	0.80	0.77
	SGD	0.9	0.58	0.58	0.494	0.376	0.689	0.668	0.725	0.66	0.63
EN Movie	BA-3+	0.99	0.99	0.896	0.87	0.00	0.00	0.024	0.058	0.81	0.80
	SGD	0.640	0.610	0.585	0.601	0.655	0.668	0.69	0.678	0.77	0.75
EN Yelp	BA-3+	0.99	0.99	0.87	0.86	0.00	0.00	0.030	0.02	0.75	0.70
	SGD	0.87	0.809	0.79	0.854	0.325	0.318	0.490	0.424	0.61	0.57

Since we aim to improve the learning capacity of RNN as good as advanced deep learning language models such as LSTMs, we compared the proposed training algorithm with the advanced neural language models, including chain-structured and tree-structured language models. For this purpose, LSTM has been modeled with the same hidden layer number and training optimizer. Tree-LSTM has been modeled with similar hyperparameters to the RNTN model, which operates over MS-TR treebank [90]. The results of the Recursive Neural Tensor Network (RNTN) model have been taken from [90]. Table 5.8 reports the hyperparameters of the advanced deep language models and Table 5.9 reports comparisons of accuracy results of advanced recurrent and recursive language models for Turkish binary classification datasets over test dataset. According to the experiments, it is observed that the RNN model combined with the BA-3+ training algorithm performed close or as good as advanced recurrent and recursive deep language models and gave comparable results.

Table 5.8 Parameter setting for LSTM and Tree-LSTM models

Parameter	**Value**
$x^{(i)} = (x_1, x_2, \ldots, x_{t-1}, x_t)$	max length = 50
embedding dimension	300
nhidden	32
epoch	100
batch size	32
optimizer	SGD
learning rate	0.01

Table 5.9 Comparisons of average accuracy results of advanced recurrent and recursive language models for Turkish binary classification datasets

Models	**Book**	**Electr.**	**DVD**	**Kitchen**	**Movie**	**Twitter**
LSTM	0.823	0.725	0.751	0.75	0.835	0.895
Tree-LSTM	0.883	0.853	0.85	0.82	0.88	0.89
RNTN	0.86	0.866	0.824	0.798	0.88	0.835
RNN BA3+	0.88	0.801	0.866	0.818	0.854	0.838
RNN SGD	0.808	0.75	0.734	0.704	0.721	0.744

The success of BA-3+ depends on its hybrid metaheuristic nature. BA-3+ evaluates only three candidate models, each having the same deep RNN architecture with different learnable parameters [165]. The training process starts with these three candidate models (number of scout bees), which dynamically search for the optimum values of the learnable parameters, and continues selecting the best model for local search to find better solutions. This is the feature of BA-3+ that brings good initial parameters for the iterative learning process. After each iteration of the proposed algorithm, if more optimum values of the

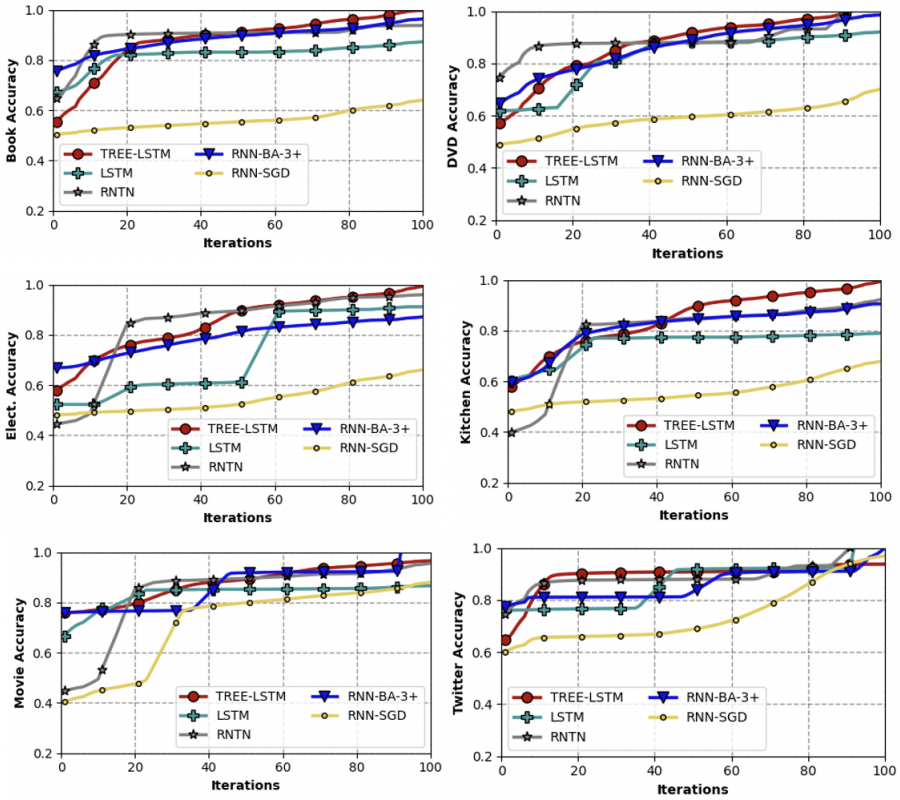

Figure 5.9 Comparison of BA-3+ performance with advanced models and RNN model trained with SGD for some datasets

learnable parameters are found, they will be updated incrementally. This feature of BA-3+ yields faster convergence. Additionally, as BA-3+ combines local SGD training with the SVD , it can exploit the learning ability of SGD while controlling VEG, making BA-3+ a hybrid meta-learning algorithm combining gradient-free and gradient-based optimisation. Finally, BA-3+ has a random exploration operator, namely, global search operator, which enables the exploration of new promising solutions while preventing the optimisation process from being trapped at local stationary points. Beside these advantages, critical hyper-parameters, such as learning rate and neighbourhood size, were empirically selected for BA-3+ as for SGD. For future studies, hyperparameter tuning would be performed to learn much larger datasets. Learning large datasets may also require the parallel running of the proposed algorithm and more powerful hardware resources.

As it can be clearly seen from Figure 5.9, RNN-SGD performed well for only one dataset. Accordingly, we can say that the performance of the systems can be better when the models are trained with huge datasets such as the Twitter classification dataset (see Table 5.4). However, in all other cases, we found that the BA algorithm performed well and yielding results just as well as advanced language models. The experimental results demonstrate that BA-3+ gives us a chance to get rid of the disadvantages of the SGD algorithm and can handle the VEG problem. Although BA-3+ time consumption is longer than SGD,

the accuracy value and F1 measure are higher for all classification datasets. Furthermore, since BA-3+ employs only three scout bees, the total training time is shorter than the DE and PSO algorithms and it outperformed the DE and PSO (see Table 5.5 and Table 5.6). BA-3+ and PSO run-times are similar, but BA-3+ gave better results in terms of accuracy. Even though the DE algorithm was tested for 100 generations, it could not reach the BA-3+ performance.

5.5 Summary

This section has described the use of the Ternary Bees Algorithm (BA-3+) as a training algorithm for finding the optimal set of parameters of a sequential deep RNN language model for the sentiment classification task. BA-3+ has been modeled as a sequential model and tested on nine different datasets including Turkish and English text. The model conducts an exploitative local search with the best bee and an explorative global search with the worst bee. The in-between bee is used for improved Stochastic Gradient Descent (SGD) learning with Singular Value Decomposition (SVD) to stabilise the updated model parameters after the application of SGD. This strategy is adopted to prevent the vanishing and exploding gradients problem of SGD training. The proposed BA-3+ algorithm guarantees fast convergence as it combines local search, global search, and SGD learning with SVD, and it is faster than other iterative, metaheuristic algorithms, as it employs only three candidate solutions in each training step. BA-3+ has been tested on the sentiment classification task with different datasets, and comparative results were obtained for chain-structured and tree-structured deep language models, Differential Evolution (DE), and Particle Swarm Optimization (PSO) algorithms. BA-3+ converged to the global minimum faster compared to the DE and PSO algorithms and it outperformed the SGD, DE, and PSO algorithms both for the Turkish and English datasets. According to the experimental results, BA-3+ performed better compared to the standard SGD training algorithm and RNN combined with BA-3+ performs as good as for advanced deep neural language models. BA-3+ converged to the global minimum faster than the DE and PSO algorithms, and it outperformed the SGD, DE, and PSO algorithms for the Turkish and English datasets. The accuracy value and F1 measure have improved at least with a 30% - 40% improvement than the standard SGD algorithm for all classification datasets. Accuracy rates in the RNN model trained with BA-3+ ranged from 80% to 90%, while the RNN trained with SGD was able to achieve between 50% and 60% for most datasets. The performance of the RNN model with BA-3+ has as good as for Tree-LSTMs and Recursive Neural Tensor Networks (RNTNs) language models, which achieved accuracy results of up to 90% for some datasets. The improved accuracy and convergence results show that BA-3+ is an efficient, stable algorithm for the complex classification task, and it can handle the vanishing and exploding gradients problem of deep RNNs. The small dif-

ferences between the training and validation results for the nine datasets have confirmed the efficiency of the proposed algorithm, with BA-3+ indeed offering better generalisation and faster convergence than the SGD algorithm.

6
RESULTS AND DISCUSSION

In this thesis, comprehensive research to supply new resources and approaches was presented for recursive deep learning models for Turkish sentiment analysis. The conclusions of the study contributed to three major interrelated research topics; creating a new sentiment treebank, modelling novel recursive deep learning architectures, and implementation of a new training algorithm.

The first part of the dissertation focus on constructing a Turkish Sentiment Treebank, which is obligatory for work with recursive deep learning models. We constructed MS-TR, a Morphologically Enriched Sentiment Treebank, which was implemented as an input for Recursive Deep Models to address compositional sentiment analysis for Turkish. MS-TR contributes to the lack of sentiment analysis resources that also can be used for the other recursive deep models for future studies. It is the first Turkish Sentiment Treebank that is fully labelled according to morphological structures of the words to infer the polarity of the inner nodes of MS-TR as positive and negative. The proposed annotation model has been done for four different levels, including morph-level, stem-level, token-level, and review-level. Morph-Level annotated MS-TR was aiming to retrieve hidden polarity of the suffixes in the morphologically rich word. To this end, tokens are morphologically analyzed, and they are parsed to their possible stem and suffixes. Using only a morphological analysis of the word cannot provide the correct polarity information for each case. Hence, in addition to the morphological annotation, we also proposed to use a polarity lexicon to capture polar words, which are root and do not have any morphological information. Stem-Level annotated MS-TR was constructed similar to morph-level annotated MS-TR. The only difference is using only stems of the words, and the suffixes (ending) of the words are eliminated from the word after morphological parsing. As a third level, we propose to annotate each token of the reviews using polar embedding spaces, which are constructed by using positive and negative datasets. The cosine similarity measure has been used to find the token-level label. If the cosine similarity of the positive most similar word and the target token is bigger than the cosine similarity of the negative most similar word and the target token, the token is labelled as positive; else, it is labelled as

negative. As the last annotation level, we proposed using only review-level annotated tree structures to construct MS-TR for comparing each annotation level's performance. In addition to the binary-labelled parse trees, the fine-grained dataset has been annotated by taking into account the morphological information of the words. The polarity distribution of the phrases, i.e. phrase-level to review-level labelling, was realized in two stages. The first stage is labelling words according to their polarity features which are detected from the morphological analysis of the words. In addition to the binary annotation, we have scored words whether they are booster words. Three different domain datasets were used, including product reviews, movie reviews, and the Turkish Natural Corpus essays for construction. Comparative results were obtained with the Recursive Neural Tensor Networks (RNTNs), which is operated over MS-TR, and conventional ML methods. Experiments proved that RNTN outperformed the baseline methods and achieved much better accuracy compared to the baseline methods, which cannot accurately capture the aggregated sentiment information.

In the second part of the thesis, advanced recursive deep learning architectures had been investigated. Even though RNTN obtained good results both for binary and fine-grained sentiment detection, results are not well enough. Hence tree-structured models have been improved using attention and memory blocks. To this end, LSTM models had been extended to Tree-LSTM, which can work over sentiment treebank while using memory blocks and attentive mechanism. A novel attentive compositional mechanism was proposed in binary Tree LSTMs (ACT-LSMTs) to improve the structural learning ability of the tree-structured LSTMs. ACT-LSTMs had been used to detect the important (more related) part of the long sentences. The main motivation was mimicking the human attention mechanism to memorize and learn the more important parts of the given sentences for a downstream task. The aim was to prevent the loss of information, which occurs due to padding operations. A comprehensive benchmark had also been aimed to compare the performances of advanced chain-structured and tree-structured language models over a SA task to decide which architecture is better. According to the experimental results, ACT-LSTM performed better in terms of fine-grained Sentiment Analysis (SA). They combined the advantages of the attention mechanism with the combined power of the TreeLSTM networks for Turkish SA, particularly to improve the performance of the structured fine-grained SA models. It had been observed that Tree-LSTMs performed better compared to chain-structured LSTMs and RNTN. RNTN also performed better than chain-LSTMs combined with various segmentation methods.

In the third part of the thesis, a new training algorithm was presented to overcome the vanishing and exploding gradients (VEG) problem, which usually observed while training models. A novel metaheuristic optimization approach was proposed for training deep RNNs for the sentiment classification task. The approach employs an enhanced ternary

Bees Algorithm (BA-3+) which maintains low time-complexity for large dataset classification problems by considering only three individual solutions in each iteration. The algorithm combines the collaborative search of three bees, performing local learning with exploitative search, SGD learning with singular value decomposition (SVD), and global learning with explorative search. Thus, the algorithm utilizes the greedy selection strategy of the local search operators of the basic Bees Algorithm to improve solutions, the stabilization strategy of SVD to handle the problem of VEG of the decision parameters, and the random global search strategy of the basic Bees Algorithm to achieve faster convergence avoiding getting trapped at local optima. BA-3+ had been used to find the optimal set of trainable parameters of the proposed deep recurrent learning architecture. The proposed algorithm had been compared for sentiment detection. According to the experimental results, the improved accuracy and convergence results showed that the proposed algorithm performed better compared to traditional SGD and BA-3+ is an efficient algorithm for training deep RNNs for complex classification tasks.

Tree-RNNs are powerful models to learn syntactic and semantic features, but they are not popular as traditional chain based RNNs due to their dependencies on the parse-tree representation of input data. Besides their compositional advantages, the construction of treebanks can be very time-consuming and processing the language getting much more complicated. Although there is a significant benefit in processing a sentence in a tree-structured recursive manner, data annotated with parse trees could be expensive to prepare, and it is needed to find the optimum tree structure of the sequence. Hence finding the best tree structure of sentences is still an open problem in NLP for future researches.

The artificial intelligence systems get dramatic successes with the contribution of the large data sets and improvements of the deep learning algorithms. The available resources and algorithms could be expanded for future Turkish studies. Algorithms could be implemented to learn the out-of-vocabulary (OOV) word embeddings that are especially useful for morphologically rich languages (MRLs).

Due to the black-box nature of the AI models and the nonlinearity of computation within the nested deep neural networks, the theory of the success behind these models still cannot be explained. That causes the lack of transparency and limited effectiveness of the AI systems and make it hard to gain trust from users. Nevertheless, instead of accepting the black-box structure of the deep learning models as a bug, it is necessary to explain the magic of how these black boxes generate information and answers.

REFERENCES

[1] B. Pang, L. Lee, S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, Association for Computational Linguistics, Jul. 2002, pp. 79–86. doi: 10.3115/1118693.1118704. [Online]. Available: https://www.aclweb.org/anthology/W02-1011.

[2] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2013, pp. 1631–1642, isbn: 9781937284978.

[3] R. Baly, H. Hajj, N. Habash, K. B. Shaban, W. El-Hajj, "A sentiment treebank and morphologically enriched recursive deep models for effective sentiment analysis in arabic," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 16, no. 4, Jul. 2017, issn: 2375-4699. doi: 10.1145/3086576. [Online]. Available: https://doi.org/10.1145/3086576.

[4] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *EMNLP 2011 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2011, pp. 151–161, isbn: 1937284115. doi: 10.1.1.224.9432.

[5] U. Eroğul, "Sentiment Analysis in Turkish," *Middle East Technical University, Master of Science*, 2009.

[6] M. Kaya, G. Fidan, I. H. Toroslu, "Sentiment analysis of Turkish political news," *Proceedings - 2012 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2012*, pp. 174–180, 2012. doi: 10.1109/WI-IAT.2012.115.

[7] Ç. Aytekin, "An Opinion Mining Task in Turkish Language: A Model for Assigning Opinions in Turkish Blogs to the Polarities," *Journalism and Mass Communication*, vol. 3, no. 3, pp. 179–198, 2013.

[8] M. Thelwall, K. Buckley, G. Paltoglou, "Sentiment strength detection for the social web," *J. Am. Soc. Inf. Sci. Technol.*, vol. 63, no. 1, pp. 163–173, Jan. 2012, issn: 1532-2882. doi: 10.1002/asi.21662. [Online]. Available: https://doi.org/10.1002/asi.21662.

[9] A. G. Vural, B. B. Cambazoglu, P. Senkul, Z. O. Tokgoz, "A Framework for Sentiment Analysis in Turkish: Application to Polarity Detection of Movie Reviews in Turkish," in *Computer and Information Sciences III: 27th International Symposium on Computer and Information Sciences*, E. Gelenbe, R. Lent, Eds., London:

Springer London, 2013, pp. 437–445. doi: https://doi.org/10.1007/978-1-4471-4594-3_45.

[10] R. Dehkharghani, Y. Saygin, "SentiTurkNet : a Turkish polarity lexicon for sentiment analysis," *Language Resources and Evaluation*, vol. 50, no. 3, pp. 667–685, 2016, issn: 1574-0218. doi: 10.1007/s10579-015-9307-6.

[11] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, K. J. Miller, "Introduction to wordnet: An on-line lexical database," *International Journal of Lexicography*, vol. 3, no. 4, pp. 235–244, 1990, issn: 09503846. doi: 10.1093/ijl/3.4.235.

[12] O. Bilgin, Ö. Çetinoğlu, K. Oflazer, "Building a Wordnet for Turkish," *Romanian Journal of Information Science and Technology*, 2004.

[13] S. Baccianella, A. Esuli, F. Sebastiani, "SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," May 2010. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2010/pdf/769_Paper.pdf.

[14] E. Cambria, C. Havasi, A. Hussain, "SenticNet 2: A semantic and affective resource for opinion mining and sentiment analysis," in *Proceedings of the 25th International Florida Artificial Intelligence Research Society Conference, FLAIRS-25*, 2012, pp. 202–207, isbn: 9781577355588.

[15] E. Cambria, D. Olsher, D. Rajagopal, "SenticNet 3: A common and commonsense knowledge base for cognition-driven sentiment analysis," *Proceedings of the National Conference on Artificial Intelligence*, vol. 2, pp. 1515–1521, 2014.

[16] O. Coban, B. Ozyer, G. T. Ozyer, "Sentiment analysis for Turkish Twitter feeds," pp. 2388–2391, 2015. doi: 10.1109/siu.2015.7130362.

[17] B. M. Ozyildirim, O. Coban, "An Empirical Study of the Extreme Learning Machine for Twitter Sentiment Analysis," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 3, no. 6, pp. 178–184, 2018, issn: 2147-6799. doi: 10.18201/ijisae.2018644774.

[18] E. Yıldırım, F. S. Çetin, G. Eryiğit, T. Temel, "The Impact of NLP on Turkish Sentiment Analysis," *TBV Journal of Computer Science and Engineering*, no. 1, pp. 44–51, 2014.

[19] C. Türkmenoğlu, A. C. Tantuğ, "Sentiment Analysis in Turkish Media," in *ICML 2014, International Conf. Mach. Learn. BeijingVolume WISDOM'14, Workshop Issues Sentim. Discov. Opin. Mining, June 25th, Beijing*, 2014.

[20] G. Yurtalan, M. Koyuncu, Ç. Turhan, "A polarity calculation approach for lexicon-based Turkish sentiment analysis," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 27, no. 2, pp. 1325–1339, 2019, issn: 13036203. doi: 10.3906/elk-1803-92.

[21] G. Eryiğit, F. S. Çetin, M. Yanık, T. Temel, İ. Çiçekli, "TURKSENT: A sentiment annotation tool for social media," in *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 131–134. [Online]. Available: https://www.aclweb.org/anthology/W13-2316.

[22] W. S. McCulloch, W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, 1943, issn: 00074985. doi: 10.1007/BF02478259.

[23] D. O. Hebb, *The Organization of Behavior*. New York: Wiley, 1949.

[24] M. L. Minsky, S. A. Papert, *Perceptrons*. Cambridge: MIT Press, 1969.

[25] H. Schütze, "Dimensions of meaning," in *Proceedings of the International Conference on Supercomputing*, 1992, isbn: 0818626305. doi: 10.1007/978-94-009-3847-2_3.

[26] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, 1990, issn: 10974571. doi: 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9.

[27] J. L. Elman, "Finding structure in time," *Cognitive Science*, 1990, issn: 03640213. doi: 10.1016/0364-0213(90)90002-E.

[28] Y. Bengio, A. Courville, P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013, issn: 01628828. doi: 10.1109/TPAMI.2013.50. arXiv: 1206.5538.

[29] M. Sahlgren, "The distributional hypothesis," *Italian Journal of Disability*, pp. 1–18, 2008. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1041938/FULLTEXT01.pdf.

[30] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, "Distributed representations ofwords and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013. arXiv: 1310.4546.

[31] R. Socher, Y. Bengio, C. D. Manning, "Deep learning for NLP (without magic)," p. 5, Jul. 2012. [Online]. Available: https://www.aclweb.org/anthology/P12-4005.

[32] R. Collobert, J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," Helsinki, Finland, Tech. Rep., 2008, pp. 160–167. doi: 10.1145/1390156.1390177. [Online]. Available: https://doi.org/10.1145/1390156.1390177.

[33] J. Pennington, R. Socher, C. D. Manning, "GloVe: Global vectors for word representation," in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014, isbn: 9781937284961. doi: 10.3115/v1/d14-1162.

[34] G. E. Hinton, *Learning distributed representations of concepts*, 1986. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.408.7684&rep=rep1&type=pdf.

[35] A. Lenci, "Distributional Models of Word Meaning," *Annual Review of Linguistics*, 2018, issn: 2333-9683. doi: 10.1146/annurev-linguistics-030514-125254.

[36] Z. S. Harris, "Distributional Structure," *WORD*, 1954, issn: 0043-7956. doi: 10.1080/00437956.1954.11659520.

[37] C. D. Manning, H. Schütze, G. Weikurn, "Foundations of Statistical Natural Language Processing," *SIGMOD Record*, 2002, issn: 01635808. doi: 10.1145/601858.601867.

[38] H. Anton, C. Rorres, *Elementary Linear Algebra, Applications Version, 9th Edition*. 2005, isbn: 0471433292.

[39] T. Shao, H. Chen, F. Cai, M. De Rijke, "Length-adaptive neural network for answer selection," in *SIGIR 2019 - Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 869–872, isbn: 9781450361729. doi: 10.1145/3331184.3331277. [Online]. Available: https://doi.org/10.1145/3331184.3331277.

[40] Y. Bengio, R. Ducharme, P. Vincent, C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003, issn: 1532-4435.

[41] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, C. Dyer, "Neural architectures for named entity recognition," in *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*, 2016, pp. 260–270, isbn: 9781941643914. [Online]. Available: https://www.aclweb.org/anthology/N16-1030.pdf.

[42] Q. Le, T. Mikolov, "Distributed representations of sentences and documents," in *31st International Conference on Machine Learning, ICML 2014*, 2014, isbn: 9781634393973. arXiv: 1405.4053.

[43] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, "Enriching Word Vectors with Subword Information," *Transactions of the Association for Computational Linguistics*, 2017, issn: 2307-387X. doi: 10.1162/tacl_a_00051. arXiv: 1607.04606.

[44] S. Hochreiter, J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, 1997, issn: 08997667. doi: 10.1162/neco.1997.9.8.1735.

[45] F. A. Gers, J. Schmidhuber, F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, 2000, issn: 08997667. doi: 10.1162/089976600300015015.

[46] R. Pascanu, T. Mikolov, Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13, Atlanta, GA, USA: JMLR.org, 2013, III–1310–III–1318.

[47] Y. Bengio, P. Simard, P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Transactions on Neural Networks*, 1994, issn: 19410093. doi: 10.1109/72.279181.

[48] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, "Gated feedback recurrent neural networks," in *32nd International Conference on Machine Learning, ICML 2015*, 2015, isbn: 9781510810587. arXiv: 1502.02367.

[49] H. Jaeger, H. Haas, "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication," *Science*, 2004, issn: 00368075. doi: 10.1126/science.1091277.

[50] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, 2017, issn: 21622388. doi: 10.1109/TNNLS.2016.2582924. arXiv: 1503.04069.

[51] J. Martens, I. Sutskever, "Learning recurrent neural networks with Hessian-free optimization," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, 2011, isbn: 9781450306195.

[52] V. Nair, G. E. Hinton, "Rectified linear units improve Restricted Boltzmann machines," in *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, 2010, isbn: 9781605589077.

[53] X. Glorot, A. Bordes, Y. Bengio, "Deep sparse rectifier neural networks," in *Journal of Machine Learning Research*, 2011.

[54] X. Glorot, Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Journal of Machine Learning Research*, 2010.

[55] D. A. Clevert, T. Unterthiner, S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016. arXiv: 1511.07289.

[56] Y. Lecun, Y. Bengio, G. Hinton, *Deep learning*, 2015. doi: 10.1038/nature14539.

[57] Q. V. Le, N. Jaitly, G. E. Hinton, *A simple way to initialize recurrent networks of rectified linear units*, 2015. arXiv: 1504.00941 [cs.NE].

[58] X. Xu, H. Ge, S. Li, "An improvement on recurrent neural network by combining convolution neural network and a simple initialization of the weights," in *Proceedings of 2016 IEEE International Conference of Online Analysis and Computing Science, ICOACS 2016*, 2016, isbn: 9781467377546. doi: 10.1109/ICOACS.2016.7563068.

[59] E. Vorontsov, C. Trabelsi, S. Kadoury, C. Pal, "On orthogonality and learning recurrent networks with long term dependencies," in *34th International Conference on Machine Learning, ICML 2017*, PMLR, 2017, pp. 3570–3578, isbn: 9781510855144. arXiv: 1702.00071.

[60] G. D. Magoulas, M. N. Vrahatis, G. S. Androulakis, "Improving the convergence of the backpropagation algorithm using learning rate adaptation methods," 7, vol. 11, Cambridge, MA, USA: MIT Press, Oct. 1999, pp. 1769–1796. doi: 10.1162/089976699300016223. [Online]. Available: https://doi.org/10.1162/089976699300016223.

[61] J. Martens, I. Sutskever, "Training deep and recurrent networks with hessian-free optimization," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7700 LECTU, pp. 479–535, 2012, issn: 16113349. doi: 10.1007/978-3-642-35289-8_27.

[62] J. Martens, "Deep learning via hessian-free optimization," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, Haifa, Israel: Omnipress, 2010, pp. 735–742, isbn: 9781605589077.

[63] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms: Second Edition*. Luniver Press, 2010, isbn: 1905986289.

[64] T. Desell, S. Clachar, J. Higgins, B. Wild, "Evolving deep recurrent neural networks using ant colony optimization," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, isbn: 9783319164670. doi: 10.1007/978-3-319-16468-7_8.

[65] Q. Kang, W. K. Liao, A. Agrawal, A. Choudhary, "A hybrid training algorithm for recurrent neural network using particle swarm optimization-based preprocessing and temporal error aggregation," in *IEEE International Conference on Data Mining Workshops, ICDMW*, 2017, isbn: 9781538614808. doi: 10.1109/ICDMW.2017.112.

[66] C. F. Juang, "A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 2004, issn: 10834419. doi: 10.1109/TSMCB.2003.818557.

[67] S. Fong, S. Deb, X. S. Yang, "How meta-heuristic algorithms contribute to deep learning in the hype of big data analytics," in *Advances in Intelligent Systems and Computing*, 2018, isbn: 9789811033728. doi: 10.1007/978-981-10-3373-5_1.

[68] P. J. Angeline, G. M. Saunders, J. B. Pollack, "An Evolutionary Algorithm that Constructs Recurrent Neural Networks," *IEEE Transactions on Neural Networks*, 1994, issn: 19410093. doi: 10.1109/72.265960.

[69] R. Eberhart, J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the International Symposium on Micro Machine and Human Science*, 1995. doi: 10.1109/mhs.1995.494215.

[70] H. W. Ge, Y. C. Liang, M. Marchese, "A modified particle swarm optimization-based dynamic recurrent neural network for identifying and controlling nonlinear systems," *Computers and Structures*, 2007, issn: 00457949. doi: 10.1016/j.compstruc.2007.03.001.

[71] P. Xiao, G. K. Venayagamoorthy, K. A. Corzine, "Combined training of recurrent neural networks with particle swarm optimization and backpropagation algorithms for impedance identification," in *Proceedings of the 2007 IEEE Swarm Intelligence Symposium, SIS 2007*, 2007, isbn: 1424407087. doi: 10.1109/SIS.2007.368020.

[72] N. Zhang, P. K. Behera, C. Williams, "Solar radiation prediction based on particle swarm optimization and evolutionary algorithm using recurrent neural networks," in *SysCon 2013 - 7th Annual IEEE International Systems Conference, Proceedings*, 2013, isbn: 9781467331067. doi: 10.1109/SysCon.2013.6549894.

[73] X. Cai, N. Zhang, G. K. Venayagamoorthy, D. C. Wunsch, "Time series prediction with recurrent neural networks trained by a hybrid PSO-EA algorithm," *Neurocomputing*, 2007, issn: 09252312. doi: 10.1016/j.neucom.2005.12.138.

[74] A. Blanco, M. Delgado, M. C. Pegalajar, "A real-coded genetic algorithm for training recurrent neural networks," *Neural Networks*, 2001, issn: 08936080. doi: 10.1016/S0893-6080(00)00081-2.

[75] M. Dorigo, G. Di Caro, "Ant colony optimization: A new meta-heuristic," in *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, 1999. doi: 10.1109/CEC.1999.782657.

[76] A. E. R. ElSaid, F. El Jamiy, J. Higgins, B. Wild, T. Desell, "Optimizing long short-term memory recurrent neural networks using ant colony optimization to predict turbine engine vibration," *Applied Soft Computing Journal*, 2018, issn: 15684946. doi: 10.1016/j.asoc.2018.09.013. arXiv: 1710.03753.

[77] B. Pang, L. Lee, "Opinion mining and sentiment analysis," *Found. Trends Inf. Retr.*, vol. 2, no. 1–2, pp. 1–135, Jan. 2008, issn: 1554-0669. doi: 10.1561/1500000011. [Online]. Available: https://doi.org/10.1561/1500000011.

[78] R. W. Picard, *Affective Computing*. Cambridge, MA, USA: MIT Press, 1997, isbn: 0262161702.

[79] R. Dehkharghani, B. Yanikoglu, Y. Saygin, K. Oflazer, "Sentiment analysis in turkish at different granularity levels," *Natural Language Engineering*, vol. 23, no. 4, pp. 535–559, 2017. doi: 10.1017/S1351324916000309.

[80] V. Hatzivassiloglou, K. R. McKeown, "Predicting the semantic orientation of adjectives," in *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, Madrid, Spain: Association for Computational Linguistics, Jul. 1997, pp. 174–181. doi: 10.3115/976909.979640. [Online]. Available: https://www.aclweb.org/anthology/P97-1023.

[81] K. Oflazer, "Turkish and its challenges for language processing," *Language Resources and Evaluation*, 2014, issn: 15728412. doi: 10.1007/s10579-014-9267-2.

[82] S. Baccianella, A. Esuli, F. Sebastiani, "SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta: European Language Resources Association (ELRA), May 2010. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2010/pdf/769_Paper.pdf.

[83] Ethnologue, *What are the top 200 most spoken languages? | Ethnologue*, 2020. [Online]. Available: https://www.ethnologue.com/guides/ethnologue200 (visited on 08/10/2020).

[84] *Turkic languages - Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Turkic_languages (visited on 06/06/2021).

[85] M. Haspelmath, M. S. Dryer, D. Gil, B. Comrie, *The World Atlas of Language Structures*. Oxford University Press, 2005, isbn: 9780199255917.

[86] B. Bickel, J. Nichols, "Inflectional synthesis of the verb," in *The World Atlas of Language Structures*, The World Atlas of Language Structures Online. Leipzig: Max Planck Institute for Evolutionary Anthropology., 2005, pp. 94–97.

[87] R. Cotterell, H. Schütze, S. Schütze, J. Eisner, "Morphological Smoothing and Extrapolation of Word Embeddings," Tech. Rep., pp. 1651–1660.

[88] K. Oflazer, "Turkish and its challenges for language processing," *Language Resources and Evaluation*, vol. 48, no. 4, pp. 639–653, 2014, issn: 15728412. doi: 10.1007/s10579-014-9267-2. [Online]. Available: https://link.springer.com/content/pdf/10.1007%2Fs10579-014-9267-2.pdf.

[89] T. Luong, R. Socher, C. Manning, "Better word representations with recursive neural networks for morphology," pp. 104–113, Aug. 2013. [Online]. Available: https://www.aclweb.org/anthology/W13-3512.

[90] S. Zeybek, E. Koc, A. Seçer, "MS-TR: A Morphologically Enriched Sentiment Treebank and Recursive Deep Models for Compositional Semantics in Turkish," *Cogent Engineering*, 2021. doi: 10.1080/23311916.2021.1893621.

[91] E. Demirtas, M. Pechenizkiy, "Cross-lingual polarity detection with machine translation," in *Proceedings of the 2nd International Workshop on Issues of Sentiment Discovery and Opinion Mining, WISDOM 2013 - Held in Conjunction with SIGKDD 2013*, 2013, isbn: 9781450323321. doi: 10.1145/2502069.2502078.

[92] U. Turk, F. Atmaca, S. B. Ozateş, A. Koksal, B. Ozturk, T. Gungor, A. Ozgur, "Turkish treebanking: Unifying and constructing efforts," in *LAW 2019 - 13th Linguistic Annotation Workshop, Proceedings of the Workshop*, 2019, isbn: 9781950737383. doi: 10.18653/v1/w19-4019.

[93] R. Baly, G. Badaro, G. El-Khoury, R. Moukalled, R. Aoun, H. Hajj, W. El-Hajj, N. Habash, K. Shaban, "A characterization study of Arabic Twitter data with a benchmarking for state-of-the-art opinion mining models," pp. 110–118, Apr. 2017. doi: 10.18653/v1/W17-1314. [Online]. Available: https://www.aclweb.org/anthology/W17-1314.

[94] H. Aarsleff, "The History of Linguistics and Professor Chomsky," *Language*, vol. 46, no. 3, pp. 570–585, Aug. 1970, issn: 00978507, 15350665. doi: 10.2307/412308. [Online]. Available: http://www.jstor.org/stable/412308.

[95] S. L. Frank, R. Bod, M. H. Christiansen, *How hierarchical is language use?* 2012. doi: 10.1098/rspb.2012.1741.

[96] C. Goller, A. Kuechler, "Learning task-dependent distributed representations by backpropagation through structure," in *IEEE International Conference on Neural Networks - Conference Proceedings*, vol. 1, IEEE, 1996, pp. 347–352. doi: 10.1109/icnn.1996.548916.

[97] J. B. Pollack, "Recursive distributed representations," *Artificial Intelligence*, 1990, issn: 00043702. doi: 10.1016/0004-3702(90)90005-K.

[98] R. Socher, C. D. C. Manning, A. Y. A. Ng, "Learning continuous phrase representations and syntactic parsing with recursive neural networks," *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, 2010.

[99] P. Le, W. Zuidema, "Compositional distributional semantics with long short term memory," in *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, Denver, Colorado: Association for Computational Linguistics, Jun. 2015, pp. 10–19. doi: 10.18653/v1/S15-1002. [Online]. Available: https://www.aclweb.org/anthology/S15-1002.

[100] L. Bottou, "From machine learning to machine reasoning: An essay," *Machine Learning*, 2014, issn: 08856125. doi: 10.1007/s10994-013-5335-x.

[101] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning representations by back-propagating errors," *Nature*, 1986, issn: 00280836. doi: 10.1038/323533a0.

[102] G. E. Hinton, R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Proceedings of the 6th International Conference on Neural Information Processing Systems*, ser. NIPS'93, Denver, Colorado: Morgan Kaufmann Publishers Inc., 1993, pp. 3–10.

[103] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–27, 2009, issn: 19358237. doi: 10.1561/2200000006.

[104] B. Pang, L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 115–124. doi: 10.3115/1219840.1219855. [Online]. Available: https://www.aclweb.org/anthology/P05-1015.

[105] J. Read, "Using emoticons to reduce dependency in machine learning techniques for sentiment classification," in *Proceedings of the ACL Student Research Workshop*, ser. ACLstudent '05, Ann Arbor, Michigan: Association for Computational Linguistics, 2005, pp. 43–48.

[106] J. Suttles, N. Ide, "Distant Supervision for Emotion Classification with Discrete Binary Values," in *Computational Linguistics and Intelligent Text Processing*, A. Gelbukh, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 121–136, isbn: 978-3-642-37256-8.

[107] J. H. Park, P. Xu, P. Fung, "PlusEmo2Vec at SemEval-2018 Task 1: Exploiting emotion knowledge from emoji and #hashtags," in *arXiv*, New Orleans, Louisiana: Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), Association for Computational Linguistics, 2018, pp. 264–272. doi: 10.18653/v1/s18-1039. eprint: 1804.08280.

[108] M. Mintz, S. Bills, R. Snow, D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Suntec, Singapore: Association for Computational Linguistics, Aug. 2009, pp. 1003–1011. [Online]. Available: https://www.aclweb.org/anthology/P09-1113.

[109] Y. Aksan, M. Aksan, A. Koltuksuz, T. Sezer, Ü. Mersinli, U. U. Demirhan, H. Yılmazer, G. Atasoy, S. Öz, İ. Yıldız, Ö. Kurtoğlu, "Construction of the Turkish national corpus (TNC)," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey: European Language Resources Association (ELRA), May 2012, pp. 3223–3227. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2012/pdf/991_Paper.pdf.

[110] A. A. Akın, M. D. Akın, "Zemberek, An Open Source Nlp Framework for Turkic Languages," *Structure*, vol. 10, pp. 1–5, 2007. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.556.69&rep=rep1&type=pdf.

[111] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, T. Mikolov, "Learning word vectors for 157 languages," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. [Online]. Available: https://www.aclweb.org/anthology/L18-1550.

[112] P. Qi, T. Dozat, Y. Zhang, C. D. Manning, "Universal Dependency parsing from scratch," in *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 160–170. doi: 10.18653/v1/K18-2016. [Online]. Available: https://www.aclweb.org/anthology/K18-2016.

[113] R. Socher, C. D. Manning, "Deep learning for NLP (without magic)," in *NAACL HLT 2013 Tutorial Abstracts*, Atlanta, Georgia: Association for Computational Linguistics, Jun. 2013, pp. 1–3. [Online]. Available: https://www.aclweb.org/anthology/N13-4001.

[114] G. Gezici, B. Yanıkoğlu, "Sentiment Analysis in Turkish," in, Oflazer K, Saraçlar M (editors). Turkish Natural Language Processing. Cham, Switzerland: Springer Publishing, 2018, pp. 255–271. doi: 10.1007/978-3-319-90165-7_12.

[115] A. Graves, "Generating Sequences With Recurrent Neural Networks," Aug. 2013. [Online]. Available: http://arxiv.org/abs/1308.0850.

[116] I. Sutskever, "Training Recurrent neural Networks," *University of Toronto, PhD Thesis*, 2013.

[117] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies*. 2001.

[118] S. R. Bowman, R. Gupta, J. Gauthier, C. D. Manning, A. Rastogi, C. Potts, "A fast unified model for parsing and sentence understanding," in *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 2016, isbn: 9781510827585. doi: 10.18653/v1/p16-1139.

[119] K. S. Tai, R. Socher, C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 1556–1566. doi: 10.3115/v1/P15-1150. [Online]. Available: https://www.aclweb.org/anthology/P15-1150.

[120] X. Zhu, P. Sobhani, H. Guo, "Long short-term memory over recursive structures," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, Lille, France: JMLR.org, 2015, pp. 1604–1612.

[121] S. Chaudhari, V. Mithal, G. Polatkan, R. Ramanath, "An attentive survey of attention models," 2020.

[122] D. Bahdanau, K. Cho, Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016.

[123] A. Parikh, O. Täckström, D. Das, J. Uszkoreit, "A decomposable attention model for natural language inference," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2249–2255. doi: 10.18653/v1/D16-1244. [Online]. Available: https://www.aclweb.org/anthology/D16-1244.

[124] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, *Attention is all you need*, 2017.

[125] T. Luong, H. Pham, C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1412–1421. doi: 10.18653/v1/D15-1166. [Online]. Available: https://www.aclweb.org/anthology/D15-1166.

[126] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach, D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, Jul. 2015, pp. 2048–2057. [Online]. Available: http://proceedings.mlr.press/v37/xuc15.html.

[127] G. Letarte, F. Paradis, P. Giguère, F. Laviolette, "Importance of self-attention for sentiment analysis," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 267–275. doi: 10.18653/v1/W18-5429. [Online]. Available: https://www.aclweb.org/anthology/W18-5429.

[128] W. Li, F. Qi, M. Tang, Z. Yu, "Bidirectional LSTM with self-attention mechanism and multi-channel features for sentiment classification," *Neurocomputing*, 2020, issn: 18728286. doi: 10.1016/j.neucom.2020.01.006.

[129] P. Qin, W. Xu, J. Guo, "Designing an adaptive attention mechanism for relation classification," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 4356–4362. doi: 10.1109/IJCNN.2017.7966407.

[130] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiský, P. Blunsom, "Reasoning about entailment with neural attention," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016. arXiv: 1509.06664.

[131] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4945–4949. doi: 10.1109/ICASSP.2016.7472618.

[132] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, J. Wu, "Sequential Recommender System based on Hierarchical Attention Networks," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, {IJCAI-18}*, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 3926–3932. doi: 10.24963/ijcai.2018/546. [Online]. Available: https://doi.org/10.24963/ijcai.2018/546.

[133] L. Zhang, P. Liu, J. A. Gulla, "Dynamic attention-integrated neural network for session-based news recommendation," *Machine Learning*, vol. 108, no. 10, pp. 1851–1875, 2019, issn: 1573-0565. doi: 10.1007/s10994-018-05777-9. [Online]. Available: https://doi.org/10.1007/s10994-018-05777-9.

[134] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018.

[135] A. Galassi, M. Lippi, P. Torroni, "Attention in natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–18, 2020. doi: 10.1109/TNNLS.2020.3019893.

[136] Y. Zhou, C. Liu, Y. Pan, "Modelling sentence pairs with tree-structured attentive encoder," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 2912–2922. [Online]. Available: https://www.aclweb.org/anthology/C16-1274.

[137] L. Dong, F. Wei, C. Tan, D. Tang, M. Zhou, K. Xu, "Adaptive Recursive Neural Network for target-dependent Twitter sentiment classification," in *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, vol. 2, Association for Computational Linguistics, 2014, pp. 49–54, isbn: 9781937284732. doi: 10.3115/v1/p14-2009. [Online]. Available: https://www.aclweb.org/anthology/P14-2009.

[138] F. Kurt, D. Kisa, P. Karagoz, "Investigating the effect of segmentation methods on neural model based sentiment analysis on informal short texts in turkish," 2019.

[139] T. Mikolov, M. Karafiát, L. Burget, C. Jan, S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*, 2010.

[140] Z. Che, S. Purushotham, K. Cho, D. Sontag, Y. Liu, "Recurrent Neural Networks for Multivariate Time Series with Missing Values," *Scientific Reports*, 2018, issn: 20452322. doi: 10.1038/s41598-018-24271-9. arXiv: 1606.01865.

[141] N. Kalchbrenner, P. Blunsom, "Recurrent continuous translation models," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1700–1709. [Online]. Available: https://www.aclweb.org/anthology/D13-1176.

[142] Q. You, H. Jin, Z. Wang, C. Fang, J. Luo, "Image captioning with semantic attention," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, isbn: 9781467388504. doi: 10.1109/CVPR.2016.503. arXiv: 1603.03925.

[143] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 6, no. 2, pp. 107–116, Apr. 1998, issn: 0218-4885. doi: 10.1142/S0218488598000094. [Online]. Available: https://doi.org/10.1142/S0218488598000094.

[144] I. Sutskever, G. Hinton, "Learning multilevel distributed representations for high-dimensional sequences," in *Journal of Machine Learning Research*, 2007.

[145] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, 2014, issn: 15337928.

[146] G. Chuangxin, Y. Gen, Z. Chengzhi, W. Xueping, C. Xiu, "SoC estimation for lithium-ion battery using recurrent NARX neural network and genetic algorithm," in *IOP Conference Series: Materials Science and Engineering*, 2019. doi: 10.1088/1757-899X/486/1/012076.

[147] K. O. Stanley, R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, 2002, issn: 10636560. doi: 10.1162/106365602320169811.

[148] A. Ororbia, A. E. R. ElSaid, T. Desell, "Investigating recurrent neural network memory structures using neuro-evolution," in *GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference*, 2019, isbn: 9781450361118. doi: 10.1145/3321707.3321795. arXiv: 1902.02390.

[149] A. Darwish, A. E. Hassanien, S. Das, "A survey of swarm and evolutionary computing approaches for deep learning," *Artificial Intelligence Review*, 2020, issn: 15737462. doi: 10.1007/s10462-019-09719-2.

[150] D. T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, M. Zaidi, "The Bees Algorithm - A Novel Tool for Complex Optimisation Problems," in *Intelligent Production Machines and Systems - 2nd I*PROMS Virtual International Conference 3-14 July 2006*, 2006, isbn: 9780080451572. doi: 10.1016/B978-008045157-2/50081-X.

[151] D. T. Pham, E. Koç, A. Ghanbarzadeh, "Optimization of the weights of multi-layered perceptions using the Bees Algorithm," no. June 2016, pp. 38–46, 2006.

[152] D. T. Pham, M. Castellani, "A comparative study of the Bees Algorithm as a tool for function optimisation," *Cogent Engineering*, 2015, issn: 23311916. doi: 10.1080/23311916.2015.1091540.

[153] A. H. Ismail, N. Hartono, S. Zeybek, D. T. Pham, "Using the Bees Algorithm to solve combinatorial optimisation problems for TSPLIB," in *IOP Conference Series: Materials Science and Engineering*, 2020. doi: 10.1088/1757-899X/847/1/012027.

[154] Y. Laili, F. Tao, D. T. Pham, Y. Wang, L. Zhang, "Robotic disassembly re-planning using a two-pointer detection strategy and a super-fast bees algorithm," *Robotics and Computer-Integrated Manufacturing*, 2019, issn: 07365845. doi: 10.1016/j.rcim.2019.04.003.

[155] D. Erhan, P. A. Manzagol, Y. Bengio, S. Bengio, P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training," in *Journal of Machine Learning Research*, 2009.

[156] S. S. Talathi, A. Vartak, "Improving performance of recurrent neural network with relu nonlinearity," 2015. arXiv: 1511.03771 [cs.NE].

[157] E. O. Wilson, "The Dance Language and Orientation of Bees. Karl von Frisch. Translated from the German edition (Berlin, 1965) by Leigh E. Chadwick. Belknap Press (Harvard University Press), Cambridge, Mass., 1967. xiv + 566 pp., illus. $15," *Science*, 1968, issn: 0036-8075. doi: 10.1126/science.159.3817.864.

[158] D. S. Wilson, "The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies By Thomas D. Seeley," *Perspectives in Biology and Medicine*, 1997, issn: 1529-8795. doi: 10.1353/pbm.1997.0016.

[159] R. L. Burden, J. D. Faires, *Numerical Analysis 9th Edition*. 2011, isbn: 9780538733519.

[160] L. N. Trefethen, D. Bau, *Numerical Linear Algebra*. 1997, isbn: 978-0-89871313-61-9.

[161] L. Bottou, "On-line learning and stochastic approximations," in *On-Line Learning in Neural Networks*, D. Saad, Ed., ser. Publications of the Newton Institute. Cambridge University Press, 1999, pp. 9–42. doi: 10.1017/CBO9780511569920.003.

[162] A. Hayran, M. Sert, "Kelime Gömme ve Füzyon Tekniklerine Dayali Mikroblog Verileri Üzerinde Duygu Analizi," in *2017 25th Signal Processing and Communications Applications Conference, SIU 2017*, 2017, isbn: 9781509064946. doi: 10.1109/SIU.2017.7960519.

[163] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015.

[164] N. Asghar, "Yelp Dataset Challenge: Review Rating Prediction," May 2016. [Online]. Available: http://arxiv.org/abs/1605.05362.

[165] S. Zeybek, D. T. Pham, E. Koç, A. Seçer, "An Improved Bees Algorithm for Training Deep Recurrent Networks for Sentiment Classification," *Symmetry*, 2021. doi: 10.3390/sym13081347.

A SOURCE CODES

A Morphologically Enriched Turkish Sentiment Treebank (MS-TR) can be downloaded from `https://data.mendeley.com/datasets/nz7vm5rchd/1`.

Related source code for Chapter 3, Chapter 4 and Chapter 5 can be downloaded from `https://github.com/sultanzeybek`.

PUBLICATIONS FROM THE THESIS

Papers

1. S. Zeybek, D. T. Pham, E. Koc, A. Secer, "An Improved Bees Algorithm for Training Deep Recurrent Networks for Sentiment Classification", Symmetry. 2021; 13(8):1347. https://doi.org/10.3390/sym13081347.

2. S. Zeybek, E. Koc, A. Secer, "MS-TR: A Morphologically Enriched Sentiment Treebank and Recursive Deep Models for Compositional Semantics in Turkish," Cogent Engineering, 2021. https://doi.org/10.1080/23311916.2021.1893621.

3. S. Zeybek, "MS-TR: A Morphologically Enriched Turkish Sentiment Treebank," Mendeley Data, vol. 1, 2020. doi: 10.17632/nz7vm5rchd.1 Available Online: https://data.mendeley.com/datasets/nz7vm5rchd/1

4. S. Zeybek, E. Koc, A. Secer, "Attentive Composition Mechanisms and Memory-Blocks over Recursive Structures for Turkish Sentiment Analysis", Applied Sciences, July 2021 (submitted).

Projects

1. **Project Title:** Training of Deep Recursive Neural Networks Using Bees Algorithm
   Supporting Institution / Programme: The Scientific and Technological Research Council of Turkey (TUBITAK), 2214/A International Research Fellowship Programme, Grant No. 1059B141800193
   Research Institution / Country: University of Birmingham, College of Engineering and Physical Sciences, Autonomous Remanufacturing Laboratory, Edgbaston, Birmingham, UK
   Period: June 2019 - June 2020