ANKARA YILDIRIM BEYAZIT UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

A NOVEL HYBRID LANGUAGE MODEL FOR FINITE STATE-BASED MORPHOLOGY

Ph.D. Thesis by

Ayla KAYABAŞ

Department of Computer Engineering

October, 2020

ANKARA

A NOVEL HYBRID LANGUAGE MODEL FOR FINITE STATE-BASED MORPHOLOGY

A Thesis Submitted to

The Graduate School of Natural and Applied Sciences of

Ankara Yıldırım Beyazıt University

In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Engineering, Department of Engineering and Natural Sciences

by

Ayla KAYABAŞ

October, 2020

ANKARA

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **A NOVEL HYBRID LANGUAGE MODEL FOR FINITE STATE-BASED MORPHOLOGY** completed by **Ayla KAYABAŞ** under the supervision of **Asst. Prof. Ahmet Ercan TOPCU** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Asst. Prof. Ahmet Ercan TOPCU

Supervisor

Asst. Prof. Özkan KILIÇ

Jury Member

Asst. Prof. Burcu CAN BUĞLALILAR

Jury Member

Prof. Dr. Fatih KOYUNCU

Jury Member

Assoc. Prof. Gazi Erkan BOSTANCI

Jury Member

Prof. Dr. Ergün ERASLAN

Director

Graduate School of Natural and Applied Sciences

ETHICAL DECLARATION

I hereby declare that, in this thesis which has been prepared in accordance with the Thesis Writing Manual of Graduate School of Natural and Applied Sciences,

- All data, information and documents are obtained in the framework of academic and ethical rules,
- All information, documents and assessments are presented in accordance with scientific ethics and morals,
- All the materials that have been utilized are fully cited and referenced,
- No change has been made on the utilized materials,
- All the works presented are original,

and in any contrary case of above statements, I accept to renounce all my legal rights.

Date: 2020, 28 October **Signature:** _____

Name & Surname: Ayla KAYABAŞ

ACKNOWLEDGMENTS

With a great pleasure, I would like to thank my doctoral advisor Dr. Ahmet E. Topcu for guiding me throughout my Ph.D. work. I would like to thank him for introducing me to the academic life with his great patience.

I would like to thank Dr. Özkan Kılıç for giving constructive suggestions on the thesis and for his continuous feedbacks throughout the thesis. This work is generally constructed also with his guidelines.

I am grateful to my assessor Dr. Burcu Can Buğlalılar for her useful comments at the meetings.

I would like to express my special thanks to Prof. Dr. Fatih Koyuncu for his immense support during this dissertation reading papers and giving constructive comments.

My great pleasure also is to know Prof. Dr. Ömer Demir, he called me as 'like a dying duck' because of my difficulties for adapting in Turkey after my return from Germany. He is my mentor for both academic life and administrative life, and anyway for all life.

I would also like to express my special thanks to my professors and both the technical and the administrative staff at AYBU.

Last but not least, my big pleasures deserve my parents for their love, and keep believing in me. They have lived with me all my embarrassments during this time. I am gratefully them for encouragement me, their patience throughout my Ph.D. work, and for providing me with a good education that led to this dissertation. It is only under the shadow of their prayers and support that I have reached this milestone. I would like to dedicate this work to You.

Ayla

A NOVEL HYBRID LANGUAGE MODEL FOR FINITE STATE-BASED MORPHOLOGY

ABSTRACT

In this dissertation, we explore the combined usage of a rule-based approach and artificial neural network-based approach in Turkish morphological analysis. We design a language-independent novel hybrid algorithm combining the rule-based X-arbitrary morphological analyzer (XMOR) and an arbitrary ANN-based approach. The usage of hybrid models including both techniques is evaluated for performance improvements. Because of the agglutinative nature of the Turkish language, the suffixation of words is essential. A good rule-based morphological analyzer covers almost all words defined in the lexicon, but big data retrieved from freely available resources, such as social media, cannot be used as lexicon entries before correcting the error. Such words are usually erroneous and should be corrected prior to morphological analysis. The proposed hybrid approach is built on the idea of the dynamic generation of an artificial neural network according to two-level phonological rules. A combination of linguistic parsing, a neural network-based error correction model, and statistical filtering is utilized to increase the coverage of pure morphological analysis. The current hybrid method combines rule-based and long short-term memory-based techniques to increase the morphological analysis performance up to 99.90 percentage for OCRd data and 99.82 percentage for social media data, which represents a new state-of-the-art morphological analysis for Turkish to the best of the author's knowledge.

Keywords: Rule-based language modeling, artificial neural network-based language modeling, hybrid approaches, TRMOR, LSTM

SONLU DURUM TABANLI MORFOLOJİ İÇİN YENİ BİR HİBRİT DİL MODELİ

ÖZ

Bu tezde, Türkçe'nin morfolojik analizinde kural tabanlı bir yaklaşım ve yapay sinir ağı (ANN) tabanlı yaklaşımın hibrit kullanımı incelenmiştir. Herhangi bir kural tabanlı XMOR morfolojik analizatör ile yapay sinir ağı tabanlı bir yaklaşımı birleştiren yeni bir hibrit algoritma geliştirilmiştir. Bu geliştirilen model dilden bağımsız ve başka bir dil için de uygulanabilir olmaktadır. Her iki tekniği içeren hibrit modellerin kullanımı, performans iyileştirmeleri açısından değerlendirilmiştir. Türkçe'nin sondan eklemeli bir dil yapısına sahip olmasından dolayı, kelimeye ek koymak gerekmektedir. İyi bir kural tabanlı morfolojik analizatör, sözlükte tanımlanan hemen hemen tüm kelimeleri kapsar, ancak sosyal medya gibi ücretsiz olarak erişilebilen kaynaklardan alınan büyük veriler düzeltilmeden önce sözlüğe kaydedilememektedirler. Bu tür kelimeler genellikle hatalı olmakta ve morfolojik analizden öncesinde düzeltilmelidirler. Önerilen hibrit yaklaşım, iki seviyeli fonolojik kurallara göre yapay bir sinir ağının dinamik üretimi fikri üzerine inşa edilmiştir. Salt morfolojik analizin kapsamını artırmak için dilsel ayrıştırma, yapay sinir ağı tabanlı bir hata düzeltme modeli ve istatistiksel filtreleme kombinasyonu kullanılmaktadır. Bu tezde geliştirilen hibrit yöntem, OCRd text verilerinin morfolojik analiz performansını yüzde 99,90'a sosyal medya verileri için de yüzde 99,82'ye yükseltmektedir. Bu şekildeki hibrit yöntem kural tabanlı ve uzun kısa vadeli bellek tabanlı teknikleri birleştirmektedir. Sonuçta bu da yazarın bilgisi dahilinde, Türkçe'nin morfolojik analizi için yeni bir son teknolojiyi temsil etmektedir.

Anahtar Kelimeler: Kural tabanlı dil modellemesi, yapay sinir ağları tabanlı dil modellemesi, karışık uygulamalar, TRMOR, LSTM

CONTENTS

NOMENCLATURES

ANN	Artificial Neural Network
LSTM	Long Short-Term Memory
XMOR	X-Arbitrary Morphological Analyzer
OCR	Optical Character Recognizing
DSPV	Defined Success Point Value
NUM_MO	Number Maximum Operation
CURRENT_LSTM_ITERATION	Current LSTM Iteration Number
SFST	Stuttgarter Finite State Tool
XML	Extensible Markup Language
BiLSTM	Bidirectional Long Short-Term Memory
MODI	Microsoft Office Document Imaging

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Smart technologies such as smart houses, smart automobiles, and human-like robots are nowadays intelligent environmental realities and fundamentally rely on natural language processing (NLP). Analyzing, generating, and understanding natural language is the most challenging field of artificial intelligence. Neural networks are used in many fields such as image recognition, speech recognition, and natural language processing. Neural network-based systems are also widely used in NLP.

A computer learns rules automatically while reading the data without programmers. These approaches are based on algorithms that learn to 'understand' language without being explicitly programmed. This is possible through the use of statistical methods, where the system starts analyzing the training set (annotated corpus) to build its own knowledge and produce its own rules.

There are two basic approaches to error correction in text documents: rule-based systems and machine learning algorithms. The system does not require a massive training corpus in rule-based systems, compared to the machine learning-based approach.

Rule-based systems are good at dealing with a specific language phenomenon. One of the most obvious disadvantages of the rule-based approach is that it requires skilled experts. A linguist or a knowledgeable engineer is needed to manually encode the rules in NLP. Here, the system may be so complex that the rules

can contradict each other. While a rule-based mechanism take rules as input, a neural-based mechanism takes as its input large data and produces rules as output from trained data.

Computational morphological analyzers can be grouped into four main types: (1) rule-based, (2) statistical, (3) neural network-based, and (4) hybrid methods. The rule-based methods consist of grammatical rules of the respective languages and are built by experts such as linguists, while neural network-based methods use a huge amount of text sources, or corpora, for building such systems. In contrast, statistical methods utilize the statistical information of words or characters in a document or corpus. Statistical approaches have been applied in diverse NLP tasks very successfully, including morphological analysis, machine translation, and text classification [1]. With a hybrid model, different approaches are combined in a pipeline to increase the system's overall performance [2]. This study is based on combining two of these approaches, namely:

- Rule-based approach: TRMOR and TRmorph
- Artificial neural network-based approach: LSTM [3]

The long short-term memory (LSTM) model utilizes optical character recognition (OCR), which provides better performance for error correction without using a language model. LSTM models show better results when used for language-independent OCR [4].

Automatically correcting words in texts within a context is a great research challenge and most of the optical character recognition (OCR) error correction tools are commercial and do not have a response for every language.

Errors can be OCR-generated vs. human-generated, orthographic, etc. [5]. Error detection and the correction of OCRd texts are challenging tasks in natural language

processing if it is not a machine printed data set but a handwritten data set. The type of error is important because techniques differ greatly in the types of error correction.

OCR precision depends on image quality (e.g., scanning resolution, noise); it is influenced negatively by poor image quality and positively by high image quality and any mismatch between the instances on which the character image classifier was trained and the conduction of the characters in the printed document (e.g., font, size, spacing). Depending on the language and the image quality of the analyzed collection, there will be a different error distribution generated by an OCR process.

The errors can be categorized as 1) real-word errors and (2) non-word errors; we will handle non-word errors. By real-word error, we mean a valid but not intended word in a sentence, which thus makes the sentence syntactically or semantically ill-formed or incorrect [6]. The real-word error is a challenging problem of every OCR systems. Most suggestion-based OCR systems are not able to solve this problem.

A non-word error is a word that is detected by the OCR system but does not really exist in any entry in the lexicon with regard to the language, whereas we understand that a real-word error is that which is correctly detected by the OCR system and also exists in the lexicon, but is grammatically wrong.

We detect the OCR errors for Turkish using a morphological analyzer, namely with the TRMOR system. Correction of errors in texts is a great challenge that is interesting for agglutinating languages such as Finnish and Turkish since such languages are morphologically rich. There are two methods for correcting errors; one uses a dictionary lookup method or N-gram matching method. Despite the known weaknesses of N-grams, N-grams remain basically the state-of-the-art [7].

1.1 Related Works

Machine-readable texts via OCR have important tasks in many fields, namely in natural language processing, artificial intelligence, pattern recognition, etc. Some practical applications with OCR systems are: (1) document data compression from document image to ASCII format, (2) language processing, (3) multi-media system design [8], (4) lip reading using machine learning algorithms, etc. These techniques automatically learn linguistic information from online text corpora, and they have been applied to a number of natural language problems throughout the last decade [9]. Linguistic context-based error correction techniques were proposed to detect and correct OCR errors with respect to their grammatical and semantic context [10]. Error detection and correction in natural language processing was performed using a finite-state automaton for Urdu [11].

The quality of documents and scanners is important for OCR systems, with poorer quality leading to more erroneous output. There are some tools to detect such errors and make suggestions for erroneously analyzed words. The first one is the ABBY FineReader tool [12] that converts a scanned TIFF document into a PDF document. This tool enables many text search operations with the converted documents. MODI (Microsoft Office Document Imaging) is another approach, available with Microsoft Office version 2003-2007. MODI is a commercial application that formats scanned documents into Word documents. Additionally, TESSERACT [13] is an open-source tool that has an OCR engine. It particularly includes options for line finding, features/classification methods, and an adaptive classifier.

Karasu and Bağlan [14] recently adapted the well-known Tesseract OCR system for Turkish. They modified various routines in the Tesseract system to work for cursive script like Nastaleeq. They reported an accuracy of 97.87% and 97.71% for 14- and 16-point font sizes, respectively.

Morphological analysis finds suffixes and their morphological tags, which can

include, for example, past tense, case, plural, and person [15]. The data that are collected from OCRd documents and social media extraction are generally examined for two main aims: the detection of erroneous words and the correction of words in various predefined forms. Two primary approaches are commonly used for implementation, including rule-based and neural network-based approaches.

There are two-level morphologies that are based on finite-state mechanisms. The PC-KIMMO system is one such mechanism, developed by [16] Antworth (1990), together with the commercial XFST Xerox tool by Beesley and Karttunen [17], and the Helsinki Finite-State Transducer Technology (HFST), implemented together with the SFST [18] among others. For Turkish as well, several morphological analyzers have been developed. Oflazer's (1994) [19] analyzer is a renowned one, developed using the PC-KIMMO mechanism. The first finite-state system was implemented for Turkish by Hankamer (1986) [20], but neither of these are freely accessible.

The SFST system is a novel implementation of this representation. This tool is also used through analyzers, namely TRMOR and TRmorph, for the Turkish language. Both of these analyzers are freely available finite-state-based morphological analyzers, and TRmorph successfully handles nearly all possible morpheme types, derivations, inflections, and compounding. In past works, SFST was applied to several other languages that differ in their morphological complexity, including SMOR for German [21].

The dictionary-based OCR error correction systems are based on a lookup dictionary for detecting and correcting errors. A dictionary-based OCR error-correction study was undertaken with separate dictionaries that were created for root words and suffixes [22]. D'hont et al. [23] developed a bidirectional LSTM model for OCR post-correction. The importance of this work lies in its creation of a corrupted text to detect and correct erroneous words at the same time. These authors

did not use any pre-annotated data, but they used a fairly limited quantity of clean data for training the model and it is not clear how they trained the model with that small amount of data because LSTM methods act with either the same training and testing data or with very big data (universal = all words of the respective language) as LSTM should know the data while testing. Saluja et al. [24] demonstrated in their work a robust LSTM model to detect OCR errors and correct them for Indic languages.

Moreover, for data-driven NLP it remains unclear how linguistic knowledge is adapted by neural network-based models or how hybrid models are built by applying the linguistic knowledge in neural architectures [25].

Hybrid approaches to NLP were primarily used from the 1980s to 2010s. Different neural network models have been suggested among these, including a hybrid approach with a recurrent neural network (RNN) and a convolutional neural network (CNN) [26]. There are hybrid approaches implemented both on the same levels and on different levels that have been used to accomplish some broad tasks in NLP. These approaches are applied, for example, at artificial neural network-based levels or rule-based levels.

The hybrid mechanisms were adopted in many areas of NLP to improve performance and coverage of tasks, for part-of-speech (POS) tagging [27], and for hybrid speech recognition, with good results from the model that Graves, Jaitly, and Mohamed [28] developed using neural network-based bidirectional LSTM (BiLSTM) and a statistical-based hidden Markov model (HMM). The hybrid approach was found to outperform machine learning-based and rule-based approaches for named entity recognition with good results [2], with the researchers combining a rule-based approach and machine learning-based approach to recognize named entities in every direction, namely “Person,” “Location,” and “Organization.”

Luong and Manning [29], meanwhile, developed a hybrid neural machine translation system to successfully complete English to Czech translation tasks. The error-tolerant recognition mechanism of Oflazer [30] corrects erroneous input word forms and analyzes them morphologically for Turkish as well as for many European languages.

Hybrid grammatical error correction [31] and a hybrid approach were applied for morphological disambiguation for Turkish by Kutlu and Cicekli [32], and they also achieved good performance. That hybrid approach developed by Kutlu and Cicekli for morphological parsing appears to be the first one for Turkish. Their hybrid morphological system applies a combination of statistical-based and rule-based approaches for disambiguation in morphological analysis. They used pre-annotated hand-tagged data with 10-fold cross-validation to evaluate their system.

The disambiguating of Turkish words has been handled with different strategies, namely as statistical dependency parsing of Turkish performed by Eryiğit and Oflazer [33], morphological disambiguation rules for Turkish developed by Yuret and Ture [34], a rule-based morphological disambiguator for Turkish implemented by Daybelge and Cicekli [35], another morphological disambiguation using recurrent neural networks among others for Turkish realized by Shen et al. (2016) [36], and, lastly, a morphology-aware network for morphological disambiguation designed by Yıldız et al. [37].

1.2 Contribution of the Thesis

The task of this dissertation is developing models and techniques to detect and correct errors in natural language. For instance, a model should be able to look at a text and detect and correct errors in it. In other words, our goal is to connect two modalities of natural language processing approaches and enable a hybrid model to increase the performance of error correction systems. In this study, we describe a

rule-based approach, TRMOR, a finite-state morphology for Turkish, and a neural network-based approach, a bidirectional LSTM (BiLSTM) model. We will build our language modeling using OCRd text documents to handle machine failures. Applying these corpora, we use TRMOR to detect and LSTM to correct errors caused by optical character recognition using TRMOR.

The bidirectional LSTM consists of two unidirectional LSTMs, one encoding the inputs and the other decoding. The output of bidirectional LSTM is then a sum of forward and backward RNN outputs. In this way, bidirectional LSTM is processing information from both preceding and following contexts [38]. The model allows an arbitrary number of layers. In our model, we only use one layer, because it is suitable for our model to generate a good performance. The activation function softmax is used to predict the probability of each character in the output sequence over the output alphabet at each time step. The ADAM optimizer is used because it gives better results for many models. The learning rate value is chosen as 0.001. The numbers of epochs used in our experiments are 100 and 50.

The model implementation of this work is written in Python with its deep learning library Keras [39] and Keras used as backend Tensorflow [40]. An important feature of LSTM-based line recognizers is their ability to achieve very high recognition accuracy without using any sophisticated, handcrafted, or specialized features and without the use of any kind of post-processing steps, such as language modeling, dictionary correction, or any other adaptation. This makes the whole process very easy to use and applicable to a wide variety of scripts and languages [41]. Here we will not only be performing a comparison, but also responding to the question of what the result will be if we add two totally different systems together, and whether we will get a better solution.

Challenges of this approach: These approaches also pose some challenges. One common criticism is that in this setting the generation of a rule-based model and

the training of the LSTM model become more difficult. In this dissertation, we propose an OCR error correction model using the TRMOR model, a finite-state based approach, and LSTM model, a neural-based approach applied in natural language.

We implement the TRMOR system using a lexicon. This system covers a large Turkish computational morphology, namely inflection, derivation, and composition, and it consists of the morpho-phonological rules and morphotactics of the agglutinating word structures.

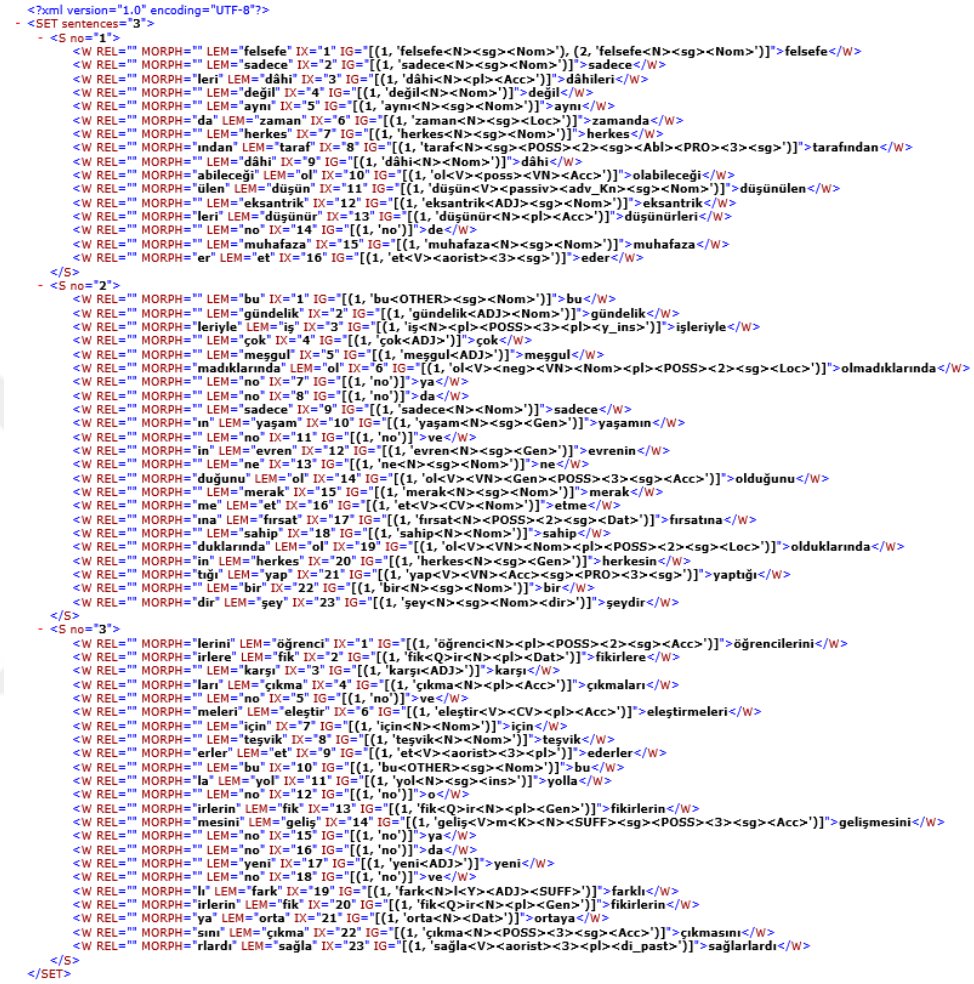
Morphological analyses are an important component of natural language processing systems, such as spelling correction systems, parsers, machine translation systems, dictionary-tools, etc. A morphological analyzer, or so-called morphological parser, starts the parsing operation by finding the root candidates [42]. The TRMOR system is a morphological parser mechanism.

We describe the morphological structure of Turkish and explain the phonological and morphological rules implemented in TRMOR. This system covers a large Turkish computational morphology as stated above. We detect erroneous words using the TRMOR approach and correct those erroneous words with the LSTM model.

Error correction system: There are two basic approaches to error correction in text documents: rule-based systems and machine learning algorithms. The rule-based system does not require large amounts of training data, while, on the other hand, the ANN-based approach does.

Error correction system processing for OCRd documents and for social media data follow a traditional step-by-step pipeline with 4 steps: (1) pre-processing of the row data, (2) detection of potential misspelling tokens, (3) correction of the erroneous data using a language model or not, (4) and development of a hybrid model of two

types of natural language processing approaches.



```
<?xml version="1.0" encoding="UTF-8"?>
- <SET sentences="3">
  - <S no="1">
      <W REL="" MORPH="" LEM="felsefe" IX="1" IG="[(1, 'felsefe<N><sg><Nom>'), (2, 'felsefe<N><sg><Nom>')]">felsefe</W>
      <W REL="" MORPH="" LEM="sadece" IX="2" IG="[(1, 'sadece<N><sg><Nom>')]">sadece</W>
      <W REL="" MORPH="leri" LEM="dâhi" IX="3" IG="[(1, 'dâhi<N><pl><Acc>')]">dâhileri</W>
      <W REL="" MORPH="" LEM="değil" IX="4" IG="[(1, 'değil<N><Nom>')]">değil</W>
      <W REL="" MORPH="" LEM="aynı" IX="5" IG="[(1, 'aynı<N><sg><Nom>')]">aynı</W>
      <W REL="" MORPH="da" LEM="zaman" IX="6" IG="[(1, 'zaman<N><sg><Loc>')]">zamanda</W>
      <W REL="" MORPH="" LEM="herkes" IX="7" IG="[(1, 'herkes<N><sg><Nom>')]">herkes</W>
      <W REL="" MORPH="ından" LEM="taraf" IX="8" IG="[(1, 'taraf<N><sg><POSS><2><sg><Abl><PRO><3><sg>')]">tarafından</W>
      <W REL="" MORPH="" LEM="dâhi" IX="9" IG="[(1, 'dâhi<N><Nom>')]">dâhi</W>
      <W REL="" MORPH="abileceği" LEM="ol" IX="10" IG="[(1, 'ol<V><poss><VN><Acc>')]">olabileceği</W>
      <W REL="" MORPH="ülen" LEM="düşün" IX="11" IG="[(1, 'düşün<V><passiv><adv_Kn><sg><Nom>')]">düşünülen</W>
      <W REL="" MORPH="" LEM="eksantrik" IX="12" IG="[(1, 'eksantrik<ADJ><sg><Nom>')]">eksantrik</W>
      <W REL="" MORPH="leri" LEM="düşünür" IX="13" IG="[(1, 'düşünür<N><pl><Acc>')]">düşünürleri</W>
      <W REL="" MORPH="" LEM="no" IX="14" IG="[(1, 'no')]">de</W>
      <W REL="" MORPH="" LEM="muhafaza" IX="15" IG="[(1, 'muhafaza<N><sg><Nom>')]">muhafaza</W>
      <W REL="" MORPH="er" LEM="et" IX="16" IG="[(1, 'et<V><aorist><3><sg>')]">eder</W>
    </S>
  - <S no="2">
      <W REL="" MORPH="" LEM="bu" IX="1" IG="[(1, 'bu<OTHER><sg><Nom>')]">bu</W>
      <W REL="" MORPH="" LEM="gündelik" IX="2" IG="[(1, 'gündelik<ADJ><Nom>')]">gündelik</W>
      <W REL="" MORPH="leriyle" LEM="iş" IX="3" IG="[(1, 'iş<N><pl><POSS><3><pl><y_ins>')]">işleriyle</W>
      <W REL="" MORPH="" LEM="çok" IX="4" IG="[(1, 'çok<ADJ>')]">çok</W>
      <W REL="" MORPH="" LEM="meşgul" IX="5" IG="[(1, 'meşgul<ADJ>')]">meşgul</W>
      <W REL="" MORPH="madıklarında" LEM="ol" IX="6" IG="[(1, 'ol<V><neg><VN><Nom><pl><POSS><2><sg><Loc>')]">olmadıklarında</W>
      <W REL="" MORPH="" LEM="no" IX="7" IG="[(1, 'no')]">ya</W>
      <W REL="" MORPH="" LEM="no" IX="8" IG="[(1, 'no')]">da</W>
      <W REL="" MORPH="" LEM="sadece" IX="9" IG="[(1, 'sadece<N><Nom>')]">sadece</W>
      <W REL="" MORPH="ın" LEM="yaşam" IX="10" IG="[(1, 'yaşam<N><sg><Gen>')]">yaşamın</W>
      <W REL="" MORPH="" LEM="no" IX="11" IG="[(1, 'no')]">ve</W>
      <W REL="" MORPH="in" LEM="evren" IX="12" IG="[(1, 'evren<N><sg><Gen>')]">evrenin</W>
      <W REL="" MORPH="" LEM="ne" IX="13" IG="[(1, 'ne<N><sg><Nom>')]">ne</W>
      <W REL="" MORPH="duğunu" LEM="ol" IX="14" IG="[(1, 'ol<V><VN><Gen><POSS><3><sg><Acc>')]">olduğunu</W>
      <W REL="" MORPH="" LEM="merak" IX="15" IG="[(1, 'merak<N><sg><Nom>')]">merak</W>
      <W REL="" MORPH="me" LEM="et" IX="16" IG="[(1, 'et<V><CV><Nom>')]">etme</W>
      <W REL="" MORPH="ına" LEM="fırsat" IX="17" IG="[(1, 'fırsat<N><POSS><2><sg><Dat>')]">fırsatına</W>
      <W REL="" MORPH="" LEM="sahip" IX="18" IG="[(1, 'sahip<N><Nom>')]">sahip</W>
      <W REL="" MORPH="duklarında" LEM="ol" IX="19" IG="[(1, 'ol<V><VN><Nom><pl><POSS><2><sg><Loc>')]">olduklarında</W>
      <W REL="" MORPH="in" LEM="herkes" IX="20" IG="[(1, 'herkes<N><sg><Gen>')]">herkesin</W>
      <W REL="" MORPH="tığı" LEM="yap" IX="21" IG="[(1, 'yap<V><VN><Acc><sg><PRO><3><sg>')]">yaptığı</W>
      <W REL="" MORPH="" LEM="bir" IX="22" IG="[(1, 'bir<N><sg><Nom>')]">bir</W>
      <W REL="" MORPH="dir" LEM="şey" IX="23" IG="[(1, 'şey<N><sg><Nom><dir>')]">şeydir</W>
    </S>
  - <S no="3">
      <W REL="" MORPH="lerini" LEM="öğrenci" IX="1" IG="[(1, 'öğrenci<N><pl><POSS><2><sg><Acc>')]">öğrencilerini</W>
      <W REL="" MORPH="irlere" LEM="fik" IX="2" IG="[(1, 'fik<Q>ir<N><pl><Dat>')]">fikirlere</W>
      <W REL="" MORPH="" LEM="karşı" IX="3" IG="[(1, 'karşı<ADJ>')]">karşı</W>
      <W REL="" MORPH="ları" LEM="çıkma" IX="4" IG="[(1, 'çıkma<N><pl><Acc>')]">çıkmaları</W>
      <W REL="" MORPH="" LEM="no" IX="5" IG="[(1, 'no')]">ve</W>
      <W REL="" MORPH="meleri" LEM="eleştir" IX="6" IG="[(1, 'eleştir<V><CV><pl><Acc>')]">eleştirmeleri</W>
      <W REL="" MORPH="" LEM="için" IX="7" IG="[(1, 'için<N><Nom>')]">için</W>
      <W REL="" MORPH="" LEM="teşvik" IX="8" IG="[(1, 'teşvik<N><Nom>')]">teşvik</W>
      <W REL="" MORPH="erler" LEM="et" IX="9" IG="[(1, 'et<V><aorist><3><pl>')]">ederler</W>
      <W REL="" MORPH="" LEM="bu" IX="10" IG="[(1, 'bu<OTHER><sg><Nom>')]">bu</W>
      <W REL="" MORPH="la" LEM="yol" IX="11" IG="[(1, 'yol<N><sg><ins>')]">yolla</W>
      <W REL="" MORPH="" LEM="no" IX="12" IG="[(1, 'no')]">o</W>
      <W REL="" MORPH="irlerin" LEM="fik" IX="13" IG="[(1, 'fik<Q>ir<N><pl><Gen>')]">fikirlerin</W>
      <W REL="" MORPH="mesini" LEM="geliş" IX="14" IG="[(1, 'geliş<V>m<K><N><SUFF><sg><POSS><3><sg><Acc>')]">gelişmesini</W>
      <W REL="" MORPH="" LEM="no" IX="15" IG="[(1, 'no')]">ya</W>
      <W REL="" MORPH="" LEM="no" IX="16" IG="[(1, 'no')]">da</W>
      <W REL="" MORPH="" LEM="yeni" IX="17" IG="[(1, 'yeni<ADJ>')]">yeni</W>
      <W REL="" MORPH="" LEM="no" IX="18" IG="[(1, 'no')]">ve</W>
      <W REL="" MORPH="lı" LEM="fark" IX="19" IG="[(1, 'fark<N>l<Y><ADJ><SUFF>')]">farklı</W>
      <W REL="" MORPH="irlerin" LEM="fik" IX="20" IG="[(1, 'fik<Q>ir<N><pl><Gen>')]">fikirlerin</W>
      <W REL="" MORPH="ya" LEM="orta" IX="21" IG="[(1, 'orta<N><Dat>')]">ortaya</W>
      <W REL="" MORPH="sını" LEM="çıkma" IX="22" IG="[(1, 'çıkma<N><POSS><3><sg><Acc>')]">çıkmasını</W>
      <W REL="" MORPH="rlardı" LEM="sağla" IX="23" IG="[(1, 'sağla<V><aorist><3><pl><di_past>')]">sağlarlardı</W>
    </S>
</SET>
```


Figure 1.1 XML integrated into the model.

Raw text is put into an annotation object and then a sequence of annotators add information in an analysis pipeline. The resulting annotation, containing all the analysis information added by the annotators, can be output in XML or plain text forms. An example of XML format is given in Figure 1.1. When creating an XML file, we deal with the overall system in a dynamic process.

XML documents are often used in information retrieval systems because the system consists of two important requirements, namely content and structural requirements

which are explicit in a formal language [43].

The OCR error correction system is organized as follows: First, we get from OCR data as output tiff data. We get OCRd documents, including books, encyclopedias, newspapers, textbooks, etc. This large corpus will be distributed to pre-processing by two separate machines. Secondly, the pre-processing step consists of the process of collecting, cleaning, and creating the corpus. By the pre-processing, we are preparing the OCR lexicon by clearing out the stop words and punctuation marks and then separating the sentences by words. Tokenization, the process of the separation of strings into words, is an important language pre-processing step for further tasks such as morphological parsing. To do this, we implement an algorithm in the Python language. Lastly, we perform morphological parsing using TRMOR, as explained in detail in the section about TRMOR, a finite-state based morphological analyzer. We will also detect the OCR errors by doing morphological analysis. After this step, we will have morphologically analyzed words and, by the TRMOR, non-analyzed words. We handle the analyzed words as correct words and put them into the correct word list. For the non-analyzed words, we will improve TRMOR.

Text corpora are usually big. It takes quite a lot of computational resources to deal with large amounts of text. This means that one wants a computer with lots of hard disk space, and lots of memory. Often, people simply use texts and regard the textual context as a surrogate for situating language in a real-world context [44]. We will build a new Turkish text corpus data set that consists of books, encyclopedias, and textbooks. This large corpus is gained by OCR scanned tiff data. The resulting entities are cleaned in two additional steps. First, stop words are identified, which have no meaning, such as ‘ve’ (and), ‘yada’ (or), ‘ama’ (but), and so on. All words that do not comply with these patterns remain. In a second step, the corpus data are cleaned by eliminating sentences or rather words that do not belong to the Turkish language. Subsequently, duplicate sentences are rejected [45]. We must make a

decision at this point as to whether we want to work token-wise or to preserve all information in the corpus data because it can contribute to meaning when a word occurs more than once.

In the tagged version of the corpus, each word is marked with a brief tag which assigns it to a specific word class. The corpus annotation, which is the process of adding interpretive information into a collection of texts, is valuable for a number of reasons, including the validation of theories of textual phenomena and the creation of corpora upon which automated learning algorithms can be trained [46].

Optical character recognition engine: The quality of the OCR engine affects the quality of the summed document data. The ScanRobot TREVENTUS 2.0 mass digitization system, which is our system for producing row data, is designed as an automatic book scanner. Its specification for books formation is 300 dpi and 400 dpi. Our scanned books involve both of the resolution formats. OCR accuracy is negatively influenced by poor image quality (e.g., scanning resolution, noise) and the formation of the characters in the printed document (e.g., font, size, and spacing). Depending on the language and the image quality of the analyzed collection, there will be a different error distribution generated by an OCR process.

Morphological analysis of a Turkish word usually results in more than one morphological parse. This ambiguity is due to the agglutinating nature of the language. Morphological disambiguation is the task of selecting the correct morphological parse for a given word in a given context [34].

Morphologically rich languages often introduce sparsity in natural language processing tasks. Whereas morphological analyzers can reduce this sparsity by analyzing morphological parsing for words, they will often introduce ambiguity by returning multiple analyses for the same surface form. The disambiguation problem between these morphological parses is further complicated by the fact that a correct

parse for a word is not only dependent on the surface form but also on other words in its context.

In this study, we present a language modeling approach to morphological disambiguation. We use TRMOR analyses that are not only word-based but also sentence-based in morphological disambiguation by presenting several LSTM-based neural architectures that encode long-range surface-level and analysis-level contextual dependencies [36]. Analyzers for morphologically complex languages often output several analysis results for the same word. The following example shows possible analysis for the word ‘kalemi’ analyzed by TRMOR, a morphological analyzer for Turkish:

kalem<N><sg><Acc>
kalem<N><sg><POSS><3><sg><Acc>
kalem<N><sg><POSS><3><sg><Nom>
kale<N><sg><POSS><1><sg><Acc>

We will use the OCRd documents and data from the TRMOR analyzed data set. We then extract a training data set from the TRMOR corpus with more than one analyzed token and disambiguate semi-automatically ambiguous Turkish words. We will train all ambiguous tokens occurring in our corpus, in contrast to the work of Shen et al. [36], Sak et al. [47], and Yildiz et al. [37]. This will be more valuable for the final outcome.

One can classify works on natural language processing (NLP) into four phases [48]. Namely, the first phase was focused on machine translation (1940-1960), while the indicator of the second phase for NLP was artificial intelligence (1960-1970), which relies much more on world knowledge and on its semantics-oriented role. The importance of the lexicon determined the third phase (1980). It was very important to investigate lexicons stimulated by the semantic-based applications. The machine-readable form of dictionaries led to the use of text corpora to validate,

improve, or adapt the initial lexical data; research was facilitated by the rapidly increasing supply of text material, After these phases, we are now currently in the fourth phase of NLP (late 1990-now), which can be identified by statistical NLP.

NLP is experiencing rapid growth as its theories and methods are deployed in a variety of new language technologies. Therefore, it is important for a wide range of researchers to work with knowledge of NLP. Within industry, this includes people who focus on human-computer interactions, business information analysis, and web software development. Within academia, it includes people in areas from bio-medicine, computational linguistics, and corpus linguistics to computer science and artificial intelligence [49].

Language processing is an important task in language processing systems, which use words as units. For agglutinating languages, it is much more difficult, since a word can appear in many different forms depending on the inflection and derivation. It is almost impossible to store any suffixed or derivative form of a word in the dictionary. Morphological analyzers are the basic components [50] of the applications to recognize words. As is the case with most other studies on the finite-state morphology of Turkish, this work is also based on the written language and not the spoken one. Apart from very few exceptions, the surface realizations of the morphemes in Turkish are conditioned by various regular morpho-phonological processes such as vowel harmony, consonant assimilation, and eradication. Morphotactics of word forms can be quite complex when multiple derivatives are affected [51].

Construction of the large Turkish corpus: We will build a large Turkish corpus, which is gained through OCRd books, and we then will improve TRMOR by applying this large corpus. With TRMOR, we will detect OCR errors. From

correctly analyzed words, we will build two separate lexicons of root words and suffixes, with the candidate root-suffix pairs of each input string [6].

Many different OCR systems are currently commercially available. Over the last five decades, machine-readable text has grown from a dream to reality. Optical character recognition has become one of the most successful applications of technology in the field of pattern recognition and artificial intelligence. Many commercial systems for performing OCR exist for a variety of applications, although the machines are still not able to compete with human reading capabilities. Since advancements in hardware have been significant since the 1990s, it is now possible to design an "ideal system" which involves "broad lexical coverage" and a lexicon as large as 100 000 words [5].

1.3 Thesis Overview

This study is organized as follows: First, we describe a language model based on neural networks in order to resolve the erroneous words within sentences. After that, we provide background information on linguistic description of the Turkish language in TRMOR and OCR error correction, and then we present the error correction methodology.

Data sources: The vast majority of modeling approaches in this area fall under the category of data-driven techniques, in which a model learns from human-annotated data. It is therefore important to highlight the available data sets and carefully study their statistics.

Corpora: Textual resources such as books, newspapers, and many other printed texts can be valuable in NLP studies. Two different types of data are used in this dissertation, with one large corpus data set for the GPU hardware testing and one small data set. The corpus used in this study relies on OCRd data, which play an

important role in language models because large amounts of data are the basis for every work, whether rule-based, statistical, neural network-based, or hybrid. Social media data [52], [53] are also used in our experiment, where rule-based and neural network-based approaches are proved. Furthermore, ANN-based LSTM models are trained from small corpora to large corpora to observe the model performances. The error rate of the 1 031 data used in k-fold cross-testing amounts to 0.8%.

The OCRd data comprise approximately 5 000 words and training the data takes 100 epochs. The total calculation time of the data is almost 16 hours for 100 epochs, and one epoch takes 9.6 minutes using a GPU. The first corpus has 1 031 tokens extracted from the existing large data set. This is also used in TRMOR to detect OCR errors in rule-based systems. The pipeline of the hybrid model is illustrated in Chapter 2 in Figure 2.1, combining the rule-based XMOR using the TRMOR and ANN-based systems using the LSTM model.

1.4 Road Map

This dissertation is divided into six parts (Introduction, Hybrid Model, Rule-based Component, Artificial Neural Network-based Component, Experimental Analysis, and Conclusion and Future Works). In the following, we briefly describe the structure of each chapter.

Chapter 2 introduces the hybrid model. We describe its architecture in Section 2.2 and propose a novel algorithm, 'Artificial Neural-Net-XMOR,' for this model in Section 2.3. The architecture of the proposed hybrid model is described and its components are given in detail. Chapter 3 explains rule-based approaches among others and gives an example approach as an explanation of a rule-based language model, namely TRMOR, a finite-state-based morphology for Turkish, in Section 3.3. Chapter 4 describes the artificial neural-based language modeling and an approach for this model, explaining long short-term memory (LSTM) in Section

4.1 and the model parameters. The experiments and the results are reported and discussed in Chapter 5. Finally, we conclude the dissertation in Chapter 6.

CHAPTER 2

A HYBRID MODEL

Hybrid systems combine several different systems in a pipeline manner. In this dissertation, we have designed a new hybrid model that utilizes a rule-based approach and a neural network-based approach together in such a pipeline process. Rule-based and neural network-based approaches both have their own unique advantages and disadvantages [54]. For example, rule-based paradigms rely on laborious work to create rules for the system. On the other hand, the training of the model plays an important role in artificial neural network (ANN)-based paradigms and it is very tedious work. Thus, in this dissertation, we present a new hybrid architecture that is significantly more successful compared to either machine-learning or rule-based systems alone.

The model basically has modules of rule-based morphological analyzers and an artificial neural network-based correction mechanism that unifies the rule-based system and ANN-based system. We used only the morphological analyzer to analyze corpus data in order to detect erroneous words. However, we need to improve our model to have a higher detection ratio. Thus, we build a new model by adapting an ANN-based system into our model that runs ANN-based and rule-based modules separately and combines the results to improve the results. In Figure 2.1, we illustrate our proposed model.

Structure: The remainder of this chapter is structured as follows: In Section 2.1, the introduction of the chapter is given. In Section 2.2, the architecture of

the model is described. The lexical resources of the model are given in Section 2.6. A general overview of different approaches is discussed for rule-based and for artificial neural-network based approaches in Section 2.4 and 2.5.

2.1 Introduction

In this work, new techniques are proposed to combine the classical rule-based mechanism and the state-of-the-art artificial neural-network-based system to increase the performance of morphological analysis.

This part of our study is based on a comparison of two previous studies, namely the studies of rule-based and ANN-based models, or, more precisely, comparison of rule-based and neural network-based systems. We explain rule-based and ANN-based models below in detail. We will not only be performing a comparison; we are also responding here to the question of what would happen if we add two totally different systems together. Would we get a better solution?

The model has modules of rule-based and of artificial neural network-based approaches, thus unifying a rule-based system (the morphological analyzers) and neural network-based system. Previously, we utilized only a rule-based finite-state-based morphological analyzer to analyze corpus data to detect erroneous words. However, we need to improve our model to obtain a higher detection ratio, and so we have built a new model by adapting an ANN-based system into it, which will now run ANN-based and rule-based systems separately and compare the results.

For time series-related and sequence-related problems, LSTMs offer a truly promising solution. A significant disadvantage, however, is that they are difficult to training. With regards to the hardware constraints, a great deal of system resources and time are used, even to train a simple model.

The most obvious disadvantage of the rule-based approach is that it requires skilled experts: it takes a linguist or a knowledge engineer to manually encode each rule in NLP. Rules must be crafted, manually, and enhanced continually. Moreover, the complexity of the system can reach a point where some of the rules can begin to contradict each other. Overall, a rule-based system can successfully capture a particular language phenomenon, wherein it can perform linguistic relationship decoding between words and sentence interpretation. Therefore, it can easily conduct sentence-level tasks, including extraction and parsing. For this reason, rule-based approaches are generally better for query analysis.

The obvious advantage of machine learning is that the model can learn from data without manually defined rules and grammar coding. On the other hand, the rule-based approaches require expert skills. In general, ANN-based approaches can significantly simplify the development of several NLP tasks when good training data sets are available. However, it is often not so easy in practice. The advantage of the implemented approaches is that they are able to be applied to documents of other scripts easily and accurately.

Rule-based Component: The finite-state-based morphological analyzer is a rule-based morphological analyzer, like other morphological analyzers that use SFST. The implementation of a morphological analyzer consists of a lexicon in which the stem morphemes are listed with their part-of-speech and inflection class, a set of regular expressions specifying morphotactics, and a set of phonological rules transforming the canonical representation of the morphemes to their surface forms. Because of the highly productive nature of agglutinative languages, the set of possible word forms cannot be listed as in the case of English; rather, it needs to be analyzed and tested [55]. We will explain this component of our hybrid model in further detail in Chapter 3.

ANN-based Component: Language models play an important role in pre- and post-processing of optical character recognition. LSTMs have been used in a wide range of problems including text recognition for images and model generations for language. The most popular of the LSTMs are the bidirectional LSTM models, which are based on the idea that the output at a particular time may not only be dependent on the previous elements in the sequence but also depend on the succeeding sequence. BiLSTM learns the error model from the erroneous words during training.

While correcting the error, there is a need for using ANN architectures such as LSTM in case multiple OCR errors occur in the long sequences of words. LSTM's ability to successfully learn on data with long-range temporal dependencies makes it a natural choice for this application.

We will explain this component of our hybrid model further in Chapter 4. We next provide a brief description of ANN-based and rule-based mechanisms and present a hybrid architecture.

2.2 The Architecture

The system components comprise two pipelined executions: a rule-based and a neural network-based approach. The execution proceeds via three main phases: (1) comparison of the paradigms; (2) filtering, or the selection and extraction of analyzed and unanalyzed words; and (3) the hybrid phase, or correction of unanalyzed erroneous words. Figure 2.1 illustrates the architecture for hybrid error correction.

Data (X) are simultaneously fed into the morphological analyzers and the ANN-based model.

1a. Rule-based component. The morphological analyzer applies a rule-based approach that parses words into their smaller units, namely suffixes. The input for this module is text data (i.e., a corpus). We have utilized two different corpora here to quantify the robustness of the overall system. One corpus is obtained from OCRd text data and the other from social media data.

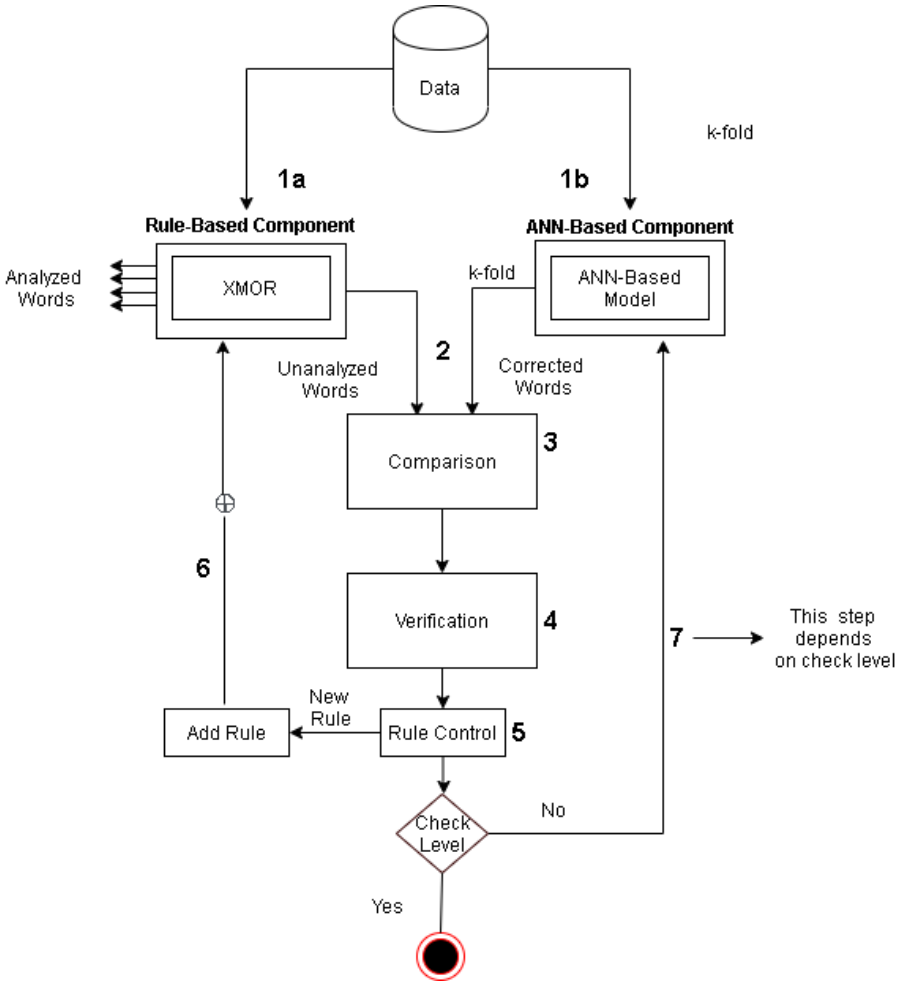


Figure 2.1 Illustration of overall hybrid model.

The output of the morphological analyzer is a list of parsed and unparsed words. These words are filtered from the same lists. The words that were successfully analyzed (A) were defined in the lexicon and processed no further. On the other hand, the unanalyzed words (Y) were marked for further analysis in the next steps.

Before the morphological process, we tokenize the corpus, whereby words from a sentence are separated and constructed from line to line as a word list. In the next step, capital letters are changed into lowercase letters, and then all punctuation is deleted from the text. After these pre-processing steps, the word list is fed into the analyzers to generate analysis. This process occurs only one time for the module.

1b. Artificial neural network-based component. The input data of this module are the same as those for the rule-based component, namely OCRd data or social media data. The data sets have two parts that are used for testing and training the ANN-based model. The quantity of training data is five times greater than that of the testing data because the model learns from training data that are free of error. We do not perform any pre-processing such as tokenization, lowercasing, or deleting of punctuation such as that done with the morphological analyzers.

The output of the ANN-based model constitutes the corrected words, which may increase the performance value of the total corrected words. The performance calculation of the ANN-based model is given in Section 4.6.1.

We use k-fold cross-validation and k is chosen as an initial value of 10, because whenever we run the ANN-based model that chooses the random initial weights it generates different corrected words depending on the performance value as shown in the third step of the algorithm by defining the k-fold value. For the evaluation of the ANN-based model, we repeat the training process to determine the minimum, average, and maximum performance values of the model. The 10-fold ANN-based model results from output text are saved for every fold running operation.

2. & 3. Comparison. During operation of the module, XMOR and ANN-based modules are executed and the results are compared. For the first time, the rule-based morphological analyzers (XMOR) explained in rule-based module and the ANN-based module were executed to compare the output of the results. In

this step, the analyzers generate two types of output, analyzed and unanalyzed words, according to their internal morphology structures. In our algorithm, the number of analyzed words is represented as Y. If the total number of words is X, the unanalyzed word count is X – Y. To improve the performance value, initially, we ran the ANN-based model with 10-fold cross-validation, and the common unanalyzed words from these analyzers and the corrected words were compared according to the lexicon of morphological systems for the training data of the ANN-based model (Figure 2.2). Comparison results are presented here as corrected.

4. Verification. The verification process is accomplished in a supervised manner; the unanalyzed words are checked to determine whether they are correct words or not and this step is repeated 10-fold for every analyzed word (Y).

5. Rule control. In this section, three primary operations are implemented:

(i) Rule control. We add the new corrected word(s) to the successfully analyzed set. However, a word may be corrected but the analyzer does not accomplish any analysis because of the lack of a rule. For the corrected word(s), the rule control module controls the rule set to make sure there is no redundancy rule. If there are new corrected words (L), then the new corrected words are added to the total analyzed words (R). The steps for the ANN-based and other modules are repeated as shown in steps 3, 4, 5, and 7 until the success value is equal to the threshold value of the overall systems for the uncorrected words from the ANN-based model and analyzers.

(ii) Increasing the total number of analyzed words (R). The analyzed total value is increased based on newly analyzed words.

(iii) Check level. Initially, k-fold iteration is completed to calculate the total analyzed words to reach a successful ending point. The results are combined using union operation as shown in Step 3a in the algorithm. Later, we operate the

ANN-based module. To eliminate the words that exist as an output of the corrected words from the ANN-based module, they are compared with the analyzed words from the analyzers.

After the process of rule control, if there is no rule for new corrected words, rules will be added to the analyzer.

6. Check level. The corrected words (Z) are combined to determine the total number of analyzed words. This operation is repeated until the defined success point value (DSPV) is reached. In later operations, in order to reach the corrected success point value, the ANN-based module is executed repeatedly. In this part, we also check the total number of iterations reached at the defined maximum iteration number, or the number of maximum operations (NUM_MO), to prevent infinite numbers of operations. This limit variable is also defined at the beginning of the algorithm.

2.3 The Algorithm

The overall system is dynamic because the initial values of variables depend on how we define the success point value.

Algorithm 1 Pseudocode of the Model

Initial Variables: The dynamic variable should be defined for calculations.

Definitions:

- RULE_BASED_XMOR_MODULE: Rule-based morphological analyzer
- ANN_BASED_MODULE: Artificial neural network-based module
- X : Total words
- Y: Analyzed words from RULE_BASED_XMOR
- DSPV : Defined Success Point Value // success value defined by the use

- NUM_MO : Number of Maximum Operations // the overall system should not run infinitely. It defines the maximum operations.
- CURRENT_ANN_BASED_MODULE_ITERATION_NUMBER: Current Iteration Number =0; //initial value is zero.

Operation A: Run ANN_BASED_MODULE(X) //find the corrected word.

1. **Run** RULE_BASED_XMOR(X) // to find Y.
2. **Find** Y = Analyzed_RULE_BASED_XMOR from step 1.
3. **Calculate** ANN_BASED_MODULE using k-fold value
   (a) **Define** k, k-fold value, the minimum number of iterations $(k \geq 10)$
   (b) **Repeat** Operation A k times to **compare** analyzed words and **find** new corrected words (z_i) i=1,2,..., k
   Z = $\{z_1 \cup z_2 \cup z_3 \cup ... \cup z_k\}$
   T = Distance (z_i, z_j) for j = i+1,..., k // Levenshtein Distance
   (c) **Until** i>k
   Calculate Analyzed_Total_Words(R) = (Y$\cup$ T)
   (d) **if** (Number_of_R(M) >= DSPV) **then**
   Terminate program
   (e) CURRENT_ANN_BASED_MODULE_ITERATION_NUMBER = k;
4. **Call** [Operation A] to get L value // L (new corrected words) is the return value of Operation A
   - **Calculate** R = R $\cup$ L
   - CURRENT_ANN_BASED_ITERATION_NUMBER = CURRENT_ANN_BASED_MODULE_ITERATION_NUMBER+1;

if (Number_of_R(M) $\geq$ DSPV) **then**

Terminate program

else

if (CURRENT_ANN_ITERATION_NUMBER $<$ NUM_MO) **then**

Go again to step 4

else

Terminate program

2.4 Rule-based Components

Morphological analyzers based on finite-state transducers are rule-based mechanisms, which consist of handcrafted rules from experts. Morphological analysis is an important component of natural language processing systems like spelling correction tools, parsers, machine translation systems, and dictionary tools.

Corpora are prepared for the morphological analyzers, namely with tokenization, lowercasing, and cleaning of the punctuation. Later, data are fed into the morphological analyzers. The TRMOR and TRmorph analyzers are explained in the following subsection.

2.4.1 *TRMOR and TRmorph Rule-Based Approaches*

In NLP, morphological analysis of text documents is an essential component, especially for agglutinative languages such as Finnish or Turkish. From information retrieval to speech recognition systems, a wide variety of NLP and understanding systems heavily depend on morphological analysis due to the fact that morphology is intertwined with other aspects of languages such as syntax and semantics. The morphological analysis basically entails breaking down a word into the smallest possible pieces. For highly inflected languages like the Turkic languages,

Hungarian, Finnish, or Korean, morphological analysis is of critical importance because the morphologies of these languages are highly intertwined with the grammar.

Phonology and morphology are the main forms of finite-state mechanisms [56]. In this research, we use two different morphological analyzers for Turkish, namely TRMOR and the well-known TRmorph. TRMOR and TRmorph are rule-based morphological analyzers, similarly to other various morphological analyzers that use SFST. The implementation of SFST tools consists of a lexicon wherein stem morphemes are given together with their part-of-speech and inflection classes, a set of regular expressions that specify the morphotactics, and a set of phonological rules that will transform the morphemes' canonical representations to surface forms. Due to the fact that agglutinative languages are extremely productive, it is infeasible to list a set of possible word forms, as could be done for English; instead, analysis and testing are required [21]. We first compared analyzers using OCR data with 1 031 words.

The morpho-phonological rules must take vowel and consonant harmony into consideration, as these patterns of harmony will impact almost all word formations in Turkish. The TRMOR components, as shown in Figure 2.2, basically consist of a lexicon and the morphotactic and morpho-phonological rules. The morphotactic word forms may be complicated if they involve multiple derivatives [51].

While designing and implementing morpho-phonological rules, the correct ordering of the rules needs to be selected carefully to ensure correct analyses. We aimed to keep the rules relatively general. Separate rules typically exist for morphemes that end in vowels and for those that end in consonants.

In Turkish word formation, suffixation is the most critical process, which entails the formation of new words with the attachment of suffix morphemes to the right side of strings. The majority of the suffix morphemes in Turkish will yield more than a

single surface realization (i.e., allomorphs) [15], and a single morpheme is capable of representing as many as 16 different forms as a result of vowel and consonant harmony, described below. As a result of this, a single morpheme may display various surface forms. Figure 2.2 illustrates the modules of the morphological analysis in TRMOR in the order of their use in an analysis process [57].

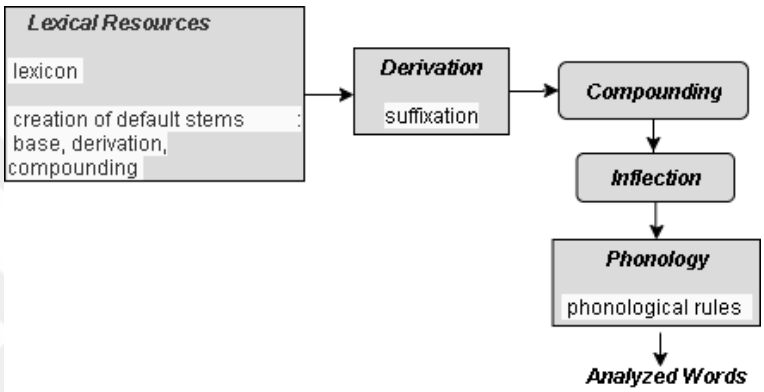


Figure 2.2 TRMOR modules.

2.5 Artificial Neural Network-based Components

In order to train the model, two types of data are required: training and testing data. The training data are clean data and they are increased by five times for training the model. For the two different types of data, the model was executed using k-fold cross-validation. One of the important characteristics of ANN-based line recognizers is their capability of achieving excellent recognition accuracy without relying on sophisticated or uniquely specialized features or any types of post-processing, such as language modeling, dictionary corrections, or other adaptations. As a result, the entire process is both easy and applicable to many different alphabets and languages [41].

2.5.1 *LSTM Networks*

While correcting errors, neural network architectures such as LSTM should be used in case multiple OCR errors occur in long sequences of words. Neural network-based approaches usually give good results when the training and the testing data are similar [4].

2.6 Lexical Resources

We constructed our own corpus data and training data for the model. The document files were prepared after OCRing the text, and they were saved as plain text. There are some special characters in the Turkish language, so we needed to change the collected data using Turkish encoding with Turkish versions ISO 8859-3, ISO 8859-9, OEM 857, and Windows-1254. Our work was performed on Linux and Windows machines using UTF-8 encoding for Linux and ISO 8859-9 for Windows.

There are two separate data sets for training and testing the ANN-based model. The training data set is labeled as clean, meaning that there are no errors in this data set. Moreover, noise data were created during the training phase to increase the error correction rates. In other words, we applied a corrupted string to our model so that it could correct the corrupted words during the testing phase. Users can also change the level of corruption to improve the model, as explained in more detail in Section 6.

Three types of data sets are used to train the model, namely test, target, and train data. The training data set is used for training models. After training the model, we evaluate it with the test data set. The test data set includes noise or unclear data. Later, the trained model generates text as the output, and we must evaluate the model to determine whether the results are credible. We test the model with

data from OCRd text documents for testing and training data without erroneous words.

The clean data from the testing data sets for each part (large/small test data) required for training the ANN-based model and the testing data set of the OCR corpus data are used for testing the ANN-based model. The words in both the training and testing data sets must exactly match, but there are errors so that the ANN-based model learns. We used clean text data to obtain artificially corrupted strings. These corrupted strings were used for training the model. The clean corpus data must be prepared at the beginning of the pre-processing of the training data. Some examples of OCR and social media non-word errors with their correct versions are as follows in Table 2.1, according to the two main categories of errors:

Table 2.1 Non-word error examples of different data types.

OCRd Data		**Social Media Data**	
Raw Data	**Ground Truth**	**Raw Data**	**Ground Truth**
anlaya- büecekleri	anlayabilecekleri	tutuomuşum	tutuyormuşum
anzalar	arızalar	şiştiii	şişti
Tann'mn	Tanrı'nın	tşk	teşekkür
1960'larm	1960'ların	bükülüo	bükülüyor
"algýnýn ötesinde'üir	"algının ötesinde"dir	lütfenn	lütfen
b.r	bir	amaliyatla	ameliyatla
a- rasmda	arasındadır	ilgineze	ilginize

i. **Non-word errors:** These are not real words, so they do not exist in a dictionary, but they are mistakenly recognized by the OCR system. For example, in Table 2.1, Tann'mn (Eng., of God) is a non-word, where the characters /r/ + /ı/ have been misrecognized as /n/.

ii. **Real-word errors:** These are usually correct words, but they are not correct in the given context. For example, the ***'çağrıştırdığım'***

("which I associated") is semantically and syntactically correct, but this word is not correct in its used context. The ***'çağrıştırdığını'*** ("her/his association") context is /n/ + /ı/ –> /m/, which is a first-person singular suffix.

2.7 Summary

This chapter explains the hybrid approaches. It covers some general concepts, including rule-based and neural-network-based approaches. First, we presented overall information according to the hybrid model. Then we discussed the strengths and weaknesses of rule-based and neural network-based approaches before presenting results in more detail.

At this point of the dissertation, we have designed a novel hybrid algorithm for the X-arbitrary morphological analyzer XMOR and an arbitrary ANN-based approach to language independently. In the next chapter, we will explain the rule-based morphological analyzer TRMOR, which is implemented for Turkish.

CHAPTER 3

RULE-BASED APPROACH: TRMOR, A FINITE-STATE-BASED MORPHOLOGICAL ANALYZER FOR TURKISH

Motivation: In this part of this dissertation, we provide some background knowledge for the material that will be addressed in the subsequent parts. We focus here on increasing the performance value of rule-based morphological analysis. We will make use of a rule-based morphological analyzer. We describe the general concepts of such analyzers, together with details on the analyzer that we will use. Moreover, we briefly discuss Turkish morphology and the basic components of the finite-state-based morphological analysis system.

3.1 Introduction

Language processing is an important task in language processing systems, which use the words as units. For agglutinating languages, it is much more difficult, since a word can appear in many different forms depending on the inflection and derivation. It is almost impossible to store any suffixed or derivative form of a word in the dictionary. Morphological analyzers are the basic components [50] of the applications to recognize words. As is the case with most other studies on the finite-state morphology of Turkish, this work is also based on the written language and not on the spoken one. Apart from very few exceptions, the surface realizations of the morphemes in Turkish are conditioned by various regular morpho-phonological processes such as vowel harmonies, consonant assimilation,

and eradication. Morphotactics of word forms can be quite complex when multiple derivatives are affected [51].

This dissertation primarily presents TRMOR, which morphologically analyzes Turkish using the Stuttgart Finite-State Transducer (SFST) tool. TRMOR is free[1] for use in academic research. Covering a great deal of Turkish morphology, it includes inflection, derivation, and some compounding. It uses a stem lexicon and rules that are both morphotactic and morpho-phonological. Herein, the Turkish morphological structure is described, the phonological and morphological rules that TRMOR implements are explained, the system is evaluated, and the special cases are tested. The evaluation of TRMOR was executed on gold-standard words. A total of 1 000 words were selected randomly from word lists in Wikipedia. For the words chosen, gold-standard analysis was achieved. TRMOR attained precision of 94.12% with the 1 000 words selected randomly from the word lists in Wikipedia. were prepared based on the gold-standard version since, to the best of our knowledge, no gold-standard segmentation exists for noncommercial purposes Turkish morphological analyzers.

Morphological analysis is the decomposition of a word into its smallest components, called morphemes. For strongly inflected languages such as Turkish, Finnish, or Hungarian, morphological analysis is very important since it identifies and categorizes the stem of the word and its class and determines its affixes. Almost all morpheme types, which are individual words in non-agglutinative languages, are included/represented in one word (in a prescribed order). This concatenation of the prepositions and pronouns into one large string leads to data sparseness. The large number of suffixes that may occur in different orders leads to a large number of possible word forms (sparse-data problems). Suffixes therefore play an important role by encoding much more grammatical information than in a configurational language, such as English.

[1] http://www.cis.uni-muenchen.de/~schmid/tools/SFST/

The agglutinative nature of the morphology is responsible for poor performance of the finite-state-based morphological analyzers [58] for languages, such as Turkish. The affixes play an important role in the structure of the language. The majority of the stems are bound and determine the class of the words in the language. The large number of word forms produce sparse-data problems, which are solved by decomposing a word into smaller, more frequent units. The reduced analysis of word forms also helps to solve sparse-data problems, statistically extracting from many word forms only the relevant features [33]. In other words, NLP techniques, such as FST, are crucial to solve the data sparseness problem with some languages, especially agglutinating languages such as Turkish.

The most important word formation process in Turkish is suffixation, which forms a new word by attaching a suffix morpheme to the right side of the string. Most Turkish suffix morphemes have more than one [59] surface realization (allomorphs). Even one morpheme can represent up to 16 different forms due to vowel and consonant harmony. Therefore, one morpheme can have different surface forms [15]. The canonical form of perfect morphemes is illustrated in Table 3.1.

The initial consonant of some suffixes and the vowels of almost all suffixes depend on the preceding consonants or vowels. The plural morpheme, for example, has two allomorphs: -lar as in *'kitaplar'* (books) and -ler as in *'kediler'* (cats). Because of the vowel harmony, the vowel of the morpheme is realized as either /a/ or /e/ depending on the backness and roundedness of the most recent vowel. The perfect morpheme, for example, has 8 forms since both the consonant and the vowel undergo variation.

In this dissertation, we implement a Turkish morphological analyzer called TRMOR using the SFST, both of which are freely available under the GNU public license. TRMOR is a two-level morphology, realized as a FST. The two-level morphology assures that the direction of analysis is reversible. SFST provides a programming

Table 3.1 Perfect suffix alternation.

dı/tı	di/ti	du/tu	dü/tü
kal-dı (s/he stayed)	gel-di (s/he came)	oku-du (s/he read)	gül-dü (s/he laughed)
kaç-tı (s/he flew)	git-ti (s/he went)	koş-tu (s/he jogged)	küs-tü (s/he was angry with)

language based on extended regular expressions, which are compiled into FSTs. TRMOR also is a rule-based morphological analyzer, like other morphological analyzers that use SFST [21]. The implementation components of TRMOR consist of a lexicon in which the stem morphemes are listed with their part-of-speech and inflection class, a set of regular expressions specifying morphotactics, and a set of phonological rules transforming the canonical representation of the morphemes to their surface forms. Because of the highly productive nature of agglutinative languages, the set of possible word forms cannot be listed as in the case of English; rather, it needs to be analyzed and tested. We evaluated TRMOR using a Turkish corpus with gold-standard morphological annotations and compared it to other morphological analyzers.

Structure: This chapter is structured as follows: The introduction section describes the state-of-the-art single-character rule-based model. We introduce some theoretical and technical background in this section. In Section 3.2, we take a look at other rule-based morphological analyzers that have been widely used in NLP applications before. In Section 3.3, we introduce TRMOR with its components. Based on that, we present a detailed description of TRMOR in Section 3.4, including examples on how it performs inflection, derivation, and compounding. We mention advantages and disadvantages of morphological analyzers before we conclude the chapter with a summary in Section 3.5.

We discuss the importance of the lexicon in Section 3.3.1. In Section 3.3.2, we explain how morphotactic processes are modeled in TRMOR. Then we briefly

present morpho-phonological processes in Section 3.3.3.

3.2 Related Works

Morphological analyzers are the base components of applications to recognize words. Turkish morphology was studied with different formalisms of finite-state techniques. The work of Jorge Hankamer [60] described a finite-state morphology and left-to-right phonology. The difference between Hankamer's work and TRMOR regarding vowel harmony will be explained in detail in Section 6.

The well-known application of finite-state systems in morphology is the two-level morphology. The two-level morphology is based on finite-state mechanisms. Some tools of these mechanisms, such the PC-KIMMO system developed by Antworth [16], the XFST: Xerox finite state tool (which is commercial) by Karttunen [17], foma: a finite-state machine toolkit and library by Huldén [61], and HFST: Helsinki Finite-State Transducer Technology, have been implemented using the SFST and OpenFst software libraries [62]. Some morphological analyzers have been developed for Turkish as well. The renowned ones were developed by Oflazer [19]. The first finite-state-based mechanism for Turkish was implemented by Hankamer [20], although neither one of these is freely accessible.

The main forms of finite-state mechanisms are phonology and morphology [56]. The state-of-the-art application of this representation is the SFST system. The system is also used by the analyzers for the Turkish language in this dissertation, namely TRMOR and TRmorph [63]. TRmorph is the first freely available finite-state-based morphological analyzer for Turkish, which in contrast to TRMOR covers almost all morpheme types, inflection, derivation, and compounding. SFST is used to build morphological analyzers for a number of other languages differing in morphological complexity. Morphological analyzers developed using SFST include a finite-state morphology for German, SMOR [21]; LATMOR for Latin

[64]; and for English, EMOR, developed by Helmut Schmid using the SFST finite-state transducer toolkit and the morphological resources of Karp et al. [65].

3.3 The Components of TRMOR

In the design of finite-state morphology, one should design and implement mechanisms that carry out the phonological alternatives of the particular language and check the validity of words for their realization.

A regular relation corresponds to a FST [66]. Implementing a morphology in SFST means incrementally building a complex regular expression, which is then compiled into a transducer. TRMOR first concatenates stem and suffix morphemes in all possible correct sequences via morphotactic rules, and then maps the resulting string to the correct surface form via morpho-phonological rules [21].

The surface level realizes the phonological form of a string through a chain of symbols in the normal, inflected form of a word. The lexical level fills the abstract basic form of input with morphotactic and lexical information, i.e. individual morphemes of a word (from entries in a lexicon). Apart from very few exceptional cases, the surface realizations of the morphemes in Turkish are conditioned through various regular morpho-phonological processes like vowel harmony, consonant assimilation, and deletion. Table 3.2 shows some surface forms as examples. Here F stands for morpheme margin, and K and Y denote triggers for vowels.

Table 3.2 Surface form examples.

Surface level	String	Analysis level
atın	at (’horse’)	at‹N›‹F›‹Y›n
çiçekte	çiçek (’flower’)	çiçek‹N›‹F›‹t›‹K›
okuyorsunuz	oku (’read’)	oku‹V›‹F›‹Y›yor ‹F›s ‹Y›n ‹Y›z

There are two major phenomena that morpho-phonological rules need to account for: vowel and consonant harmony. These influence most Turkish word formations. The lexicon and the morphotactic and morpho-phonological rules will be discussed in the next sections. The morphotactic word forms can be quite complex when multiple derivatives are involved [51]. The morphotactical order for verbal suffixes that we use in TRMOR is given in Figure 3.1.

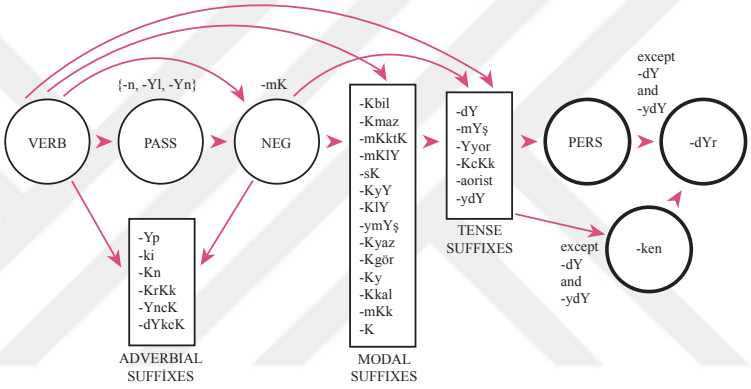


Figure 3.1 Illustration of morphotactic orders for verbs used in TRMOR.

3.3.1 *Lexicon*

A lexicon is an additional file in the SFST. It is created manually by inputting words for checking. The TRMOR lexicon contains 36,903 entries. Each entry consists of a morpheme with information about the morpheme type, the part of speech, and the inflection class. The distribution of word classes in TRMOR is presented in Table 3.3. The lexicon entries in TRMOR encode features of words (see Table 3.4). The system includes rules, which derive derivation and compounding stems from base stems [21]. Each base stem belongs to a word class and an inflectional class, which produces the inflectional endings for each case, number, gender (however, note that there is no grammatical gender in Turkish and therefore no gender encoding in

TRMOR), and person variation of the lemma [57].

Table 3.3 Distributions of word classes.

Word class	Total
N	23 214
V	2638
ADJ	1517
NE	9534

For rule-based word formation, the inflection class plays an important role. The

Table 3.4 Lexicon entries of TRMOR.

Types	Tags
Entry types	‹Stem›‹Suffix›
Stem types	‹base›‹deriv›‹compound›
Word types	‹V›‹ADJ›‹N›‹NE›
Inflectional classes	‹NomReg›‹VerbReg›...

nouns in the lexicon have 4 inflection classes, namely ‹NomReg-p›, ‹NomReg›, ‹NomReg-su›, and ‹NomRegCop›. ‹NomReg-p› occurs 16 419 times more than ‹NomReg› at 6 751. As we noted above, the ‹NomReg-p› inflection class comprises nouns that end with consonants. The inflection classes comprise the verbal and nominal categories (nouns, adjectives, adverbs). These differ in their root words' endings again. Some nominal lexicon entries with their morpheme types and inflection classes are listed in Table 3.5. In order to obtain an inflected word form, we apply an inflectional suffix to a base stem, such as oku+yor $\rightarrow$ okuyor.

For the first lexicon entry, the word *silgi* ('eraser') is analyzed. Its word class is assigned as noun (‹N›), its stem type as base stem, its origin as native, and its inflection class as ‹NomReg›. ‹NomReg› is used as a trigger, which at the same time assigns the variable $NomReg$. The second and fourth expressions are for the lexicon entries *kalem* ('pencil') and *ev* ('house'), with their lemmas,

Table 3.5 Example entries from the lexicon with their glosses.

Entry	Lemma	Gloss	Part of speech	Type	Origin	Inflection class
Stem	silgi	eraser	N	base	native	NomReg
Stem	kalem	pencil	N	base	native	NomReg-p
Stem	su	water	N	base	native	NomReg-su
Stem	ev	house	N	base	native	NomRegCop
Stem	Mehmet	Mehmet	NE	base	native	NEReg

their part-of-speech tags (in these examples ‹N› for nouns), their origin (native), and their inflection class (‹NomReg-p›). For the fourth lexicon entry defined with *ev*, which belongs to inflection class ‹NomRegCop›, all nouns can be analyzed as copulas.

We distinguish between the nouns that end with vowels and consonants in their inflection class. As one can see from the lexicon entries *kalem* and *silgi*, the second lexicon entry belongs to a different inflection class. The third lexicon entry, *su* ('water'), differs from other nouns in declination and case, and since there are many words ending in *su*, we treat it as a separate lexicon entry. Therefore, this word has a distinct inflection class. The fifth lexicon entry is a personal name and belongs to the inflection class of named entities.

The derivational suffixes define the features of a derived word form, namely the word class, the inflectional class, and the origin. These features follow the suffix morpheme in the lexicon entry of the suffix [21]. The concept of derivation is as usual in the following way:

‹stem›word ‹word class›‹stem type›‹origin›
‹affixtyp›‹simplex›‹wordclass›‹stem type›‹origin›‹suffix›‹word class›‹stem type›
‹origin›‹flexion class›

The representation of noun to verb (1) and noun-to-noun (2) derivations in TRMOR are shown as follows in the lexicon:
(1) güzel+leş → güzelleş (N → V)

‹stem›güzel ‹N›‹deriv›‹native›

‹suffix›‹simplex›‹N›‹deriv›‹native›l‹K›ş ‹V›‹base›‹native›‹VerbReg-Consonant›

(2) güzel+lik → güzellik (N → N)

‹stem›güzel ‹N›‹deriv›‹native›

‹suffix›‹simplex›‹N›‹deriv›‹native›l‹Y›k‹N›‹base›‹native›‹NomReg-p›

3.3.2 *Morphotactic Process in TRMOR*

Derivational suffixes change the part of speech while inflectional suffixes represent features such as number and case. In TRMOR, word formation happens through the concatenation of bound morphemes with other bound morphemes, such as derivational or inflectional morphemes. Free morphemes, such as -mı/mi, build yes/no questions [59]. The entry types consist of stems and suffixes. The stem types of TRMOR are composed of base stems, with the derivation participle, or as in the example *yapıyor idiysen* ('If (as you imply) you did'). However, this suffix, –idiysen, separated into morphemes as idi (=perfect of to be) + yse (cond. causal) + n (2sg), usually attaches to tense suffixes, e.g., -Yyor, -mYş, -dY, and becomes *yapıyorduysan* ('If (as you imply) you were doing') in this case. An interesting example of word formation in Turkish is given in Table 3.6 with the following example:

ölümsüzleştirilemeyeceklerdenmişim ('I am not the one of those who never converted to be immortalized').

Table 3.6 An illustration of Turkish suffixation.

root	1	2	3	4	5	6	7	8	9	10	11	12	13
öl	üm	süz	leş	tir	il	e	me	y	ecek	ler	den	miş	im
die	D_N	D_ADJ	D_V	caus	pass	opt	neg	inter-fix	fut	pl	abl	past	1sg

This extremely complex example, concatenated with 12 suffixes (except the inter-fix y in position 8), is a prototypical illustration of suffixation from the derivational

level (see Table 3.7 for some derivational suffixes implemented in TRMOR) to the inflectional level in Turkish. This type of suffixation may be encountered in the daily use of language. Here, from the verb *öl* ('die'), with the first derivational suffix -Ym (here -üm because of vowel harmony, as addressed in Section 3.3.3), it becomes a noun; with the second derivational suffix –sYz (=without) (here Y becomes /ü/), the first derived word *ölüm* ('dead') is converted into an adjective, *ölümsüz* ('immortal'); and now from the adjective, with the third derivational suffix -lKş, it again becomes a verb *ölümsüzleş* ('become immortal').

In Table 3.7, some examples of derivational suffixes implemented in TRMOR are given with their derived directions, examples, and glosses.

Table 3.7 Some derivational suffixes with examples and their glosses.

Derivational suffix	Derivation direction	Example Turkish	Gloss
-dYk	V $\rightarrow$ *ADJ*	gittiğim	that I had gone
-YcY	N $\rightarrow$ *N*	kalıcı	staying
-cK	N $\rightarrow$ *N*	Fince	Finnish
-cY	N $\rightarrow$ *N*	Türkçeci	Turkish teacher

TRMOR recognizes the productive and a few unproductive derivational suffixes. Dealing with all unproductive suffixes would be too much work. Most of the Turkish derivations are usually formed by suffixation. Some derivational suffixes are inflectional too, or vice versa. For example, the -Yn suffix can occur as a passive suffix, such as in the word *kalınır* ('one stays'), and as a derivational suffix, such as in the word *basın*, from *basmak* ('publish'), *bas+Yn* ('press'). We defined this suffix once as derivation in the lexicon and implemented it for a passive suffix as a phonological rule. It is defined only as inflectional passive suffix +Yn. For the word *basın* ('press'), there is a lexicon entry for the derivational suffix +Yn to prevent over-segmentation. There are different morphemes with the same surface forms in Turkish. We treated such morphemes differently in the lexicon

as explained in the implementation section. For semantic reasons, most suffixes in Turkish, especially the derivational ones, can be attached to only certain stems [67]. The inflection classes of suffixes differ also according to their ending in TRMOR. The vowel-ending suffixes belong to the $VerbReg - p$ inflection subclass and the consonant-ending suffixes to the $VerbReg - Ym$ inflectional subclass. These subclasses define the personal pronouns. We will briefly look at the inflection classes for the lexicon entry *belirle* ('determine') in Figure 3.2:

‹Stem›belirle‹V›‹base›‹native›‹VerbReg›

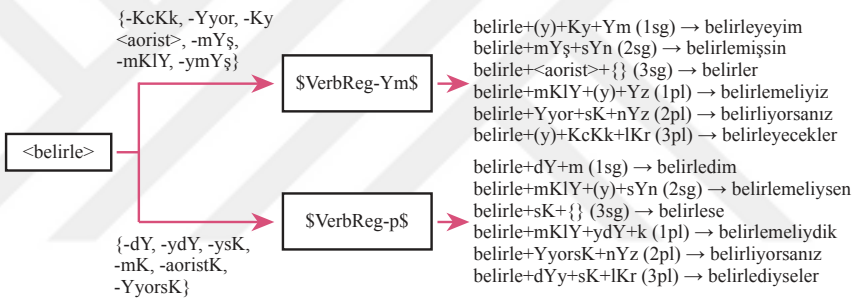


Figure 3.2 Visualization of the inflectional variants of person suffixes.

We list all possible inflectional surface variants for *belirle* based on the inflection class ‹VerbReg›. The variable ‹VerbReg› can go into two suffix inflection classes. The main inflection class can be used without damaging the morphotactical order first divided into two inflection classes because of vowel harmony. Each class contains information about inflectional endings, which determine the inflection class of person suffixes regarding the preceding suffix. Visualizations of the inflectional variants of person suffixes after some other suffixes are implemented in TRMOR using the inflection class ‹VerbReg-p› and ‹VerbReg-Ym› using the verb *belirle* (= 'determine').

3.3.3 *Morpho-phonological Process in TRMOR*

There are two styles [56] of implementing morpho-phonological rules, parallel and sequential, according to which the TRMOR transducer is built. The rules of our system are as follows. When a compiler reads the lexicon file, each line is converted to a transducer and then transducers are combined with the OR operator. The following presentation executes the rules sequentially one at a time.

“lexicon" $\| R_1 \| R_2 \| R_3 \| R_4 \| R_5 \| ... \| R_n$

Dictionaries and rules are compiled together into a finite-state transducer. Different types of linguistic characteristics can be realized by FST [68]. In sequential systems, the surface form of the lexical form is derived using ordered rules through a series of intervening representations. In a parallel description, the rules directly restrict the execution of the lexical form without access to an intermediate phase. Although it could be the case that a parallel rule takes precedence over another, it does not mean that their application is temporarily ordered.

In two-level morphology, all morpho-phonological rules are applied in parallel with identical input and output strings, and they produce the surface string in one step. The parallel rules may interact with each other in complicated ways. The other style applies the rules in sequence [56]. Each rule receives the output of the previous rule as an input and sends its output to the next rule. The sequence of rules generates multiple intermediate representations before producing the final surface form. The compiler later collapses the sequence of processing steps into a single step, which directly generates the surface form from the analysis string (in generation mode) or vice versa (in analysis mode). TRMOR uses 44 rules to map morpheme sequences to surface forms. The inflectional and derivational suffixes contain the special trigger symbols Y and K (instead of vowels), which undergo harmonization. Table 3.8 and Table 3.9 show to which vowels these trigger symbols are converted, depending on the preceding vowel, which forms the context.

During the design and implementation of the morpho-phonological rules, it is important to choose the correct order of the rules in order to avoid incorrect analyses. We tried to make the rules as general as possible. Usually there are separate rules for morphemes ending in a vowel and for morphemes ending in a consonant.

Table 3.8 The vowel harmony rule for Y.

Trigger/Context	aı	ei	ou	öü
Y	ı	i	u	ü

Table 3.9 The vowel harmony rule for K.

Trigger/Context	aıuo	eiöü
K	a	e

Vowel Harmony

Vowel harmony, a characteristic feature of Turkish morphology, states that all vowels must match within a domain (usually the non-compound word) in terms of one or more properties [69]. Figure 3.3 shows how the harmony rules transform the trigger symbols ‹K› and ‹Y› to the correct vowels depending on the previous vowel [70]. The harmony rules are implemented in TRMOR by first mapping each trigger symbol to all of its possible realizations and then filtering out vowel sequences that violate the vowel harmony constraints. The resulting word form is *koşuyorduysanız* (‘if (as you imply) you were running’) and the analysis level is as follows:

koş‹V›‹praes›‹di_past›‹cond_cop›‹2›‹pl›

A special phenomenon occurs with the Turkish present tense morpheme -iyor/-ıyor in the case of polysyllabic verb roots that end in a vowel. The final stem vowel here

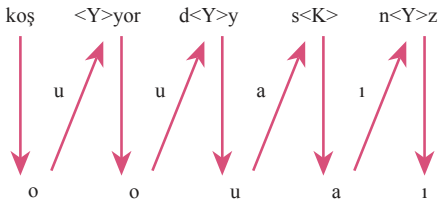


Figure 3.3 Stepwise vowel harmony.

is deleted before the vowel harmony rules are applied. For the verb *oyna* ('play'), for example, we obtain the following:

oyna + *‹Y›yor* + *‹Y›*
oyn + *‹Y›yor* + *‹Y›*
oyn + *uyor* + *‹Y›m*
oyn + *uyor* + *um*
oynuyorum

The preceding rounded back vowel /o/ triggers the realization of Y as /u/. If we applied vowel harmony first, the preceding 'a' would instead trigger the incorrect realization of Y as /ı/. The following example shows a different approach to vowel harmony, as proposed by Hankamer [60]:

Stem → öde + Yyor
Vowel deletion of suffix → öde + yor
Vowel raising → ödi + yor
Rounding → ödü + yor
Surface form → ödüyor

Hankamer first deletes the initial vowel of the suffix -Yyor and then applies vowel harmony rules to the last vowel of the stem, which is raised and rounded.

öde+yKcKk ⇒ ödi+yecek ⇒ ∗ ödüyecek
söyle+yKcKk ⇒ söyli+yecek ⇒ *söylüyecek

In contrast, our rules produce the correct results:

öde + KcKk ⇒ öde + (y)+ ecek ⇒ ödeyecek

For the suffix KcKk, no vowel is deleted. The vowel harmony rules replace the trigger symbols K with /e/ and finally the joint element 'y' is inserted. It seems that our harmony rule approach is simpler and more reliable than Hankamer's, as the example shows [60]. His approach is simple only for the two verbs *de* ('say') and *ye* ('eat'), handled as exceptions and defined as a rule in TRMOR. These verbs are highly irregular in their inflection and require special treatment. When the present tense morpheme -Yyor is added, the single vowel in the stem is deleted after applying the vowel harmony rule to Y, resulting in the form. In the case of the suffix -KcKk, a joint element /y/ is inserted after applying the vowel harmony rule for K, resulting in the form.

The vowel harmony rules for polysyllabic verbs first check which types of vowels occur in the root. If there is a low and a high vowel (see Table 3.8), then the last vowel is deleted and the -Yyor (present suffix) is chosen according to the remaining high vowel. For example, the verb *zıpla* ('jump') has a high vowel /ı/ and a low vowel /a/. /a/ is deleted, and Y is realized as the high vowel /ı/ because of the remaining vowel /ı/. Zıpla+Yyor becomes *zıplıyor* ('he/she is jumping'). Y represents the high vowels. When there exist two vowels (in polysyllabic cases, one considers only the first occurring vowels), there is a matter of low vowels (see above in Table 3.8), and Y takes the high vowel of the rounded vowel.

Consonant Harmony

If the stem ends in a voiced consonant and the initial consonant of the suffix is an alternating consonant (d/t, c/ç, g/k), then the initial consonant of the suffix becomes

voiced (d, c, g); otherwise, it is unvoiced. In the word form *kırda* (‘on the field’), for example, the last stem phoneme /r/ is voiced and the suffix begins with an alternating consonant. According to the rule, the voiced variant of the ablative suffix -da is realized.

Final Devoicing

The voiced consonants /b/, /d/, and /g/ become voiceless at the end of the word: for example, the singular form of the noun stem *kitap* (‘book’) with accusative suffix (-Y) kitap+Y, *kitabı*. However, this does not always work with monosyllabic words, such as, for example, the word *hap* (‘tablet’). Namely, hap+ı remains as *‘hapı’*, not *habı*, like *haç* (‘crucifix’) to *haçı*, *saç* (‘hair’) to *saçı*, *kat* (‘floor’) to *katı*, etc.

Dropping Vowels

Some Turkish words drop the last vowel when a suffix starting with a vowel is added. The accusative form of the word stem *akıl* (‘intelligence’), for instance, is *aklı*. Whether the vowels are dropped depends on the stem and is not predictable. In the morpheme lexicon, we therefore add a trigger symbol at the point where the deletion will take place, and we define a phonological deletion rule, which is triggered when a suffix that begins with a vowel is added.

Dropping the Plural Suffix

The plural suffix -l‹K›r (3.pl.) is completely dropped if it follows a possessive suffix -l‹K›r‹Y›.

Consonant Duplication

The final consonant of a morpheme is duplicated if it is directly preceded by a vowel. For example, the word *hak* (‘right’) becomes *hakkı* when the accusative suffix -Y is added, while the trigger ‘X’ is assigned for the duplication. In Figure

3.4, one can see the other constraints of the word *hak*, namely in the forms of plural and singular, again in accusative, dative, and genitive.

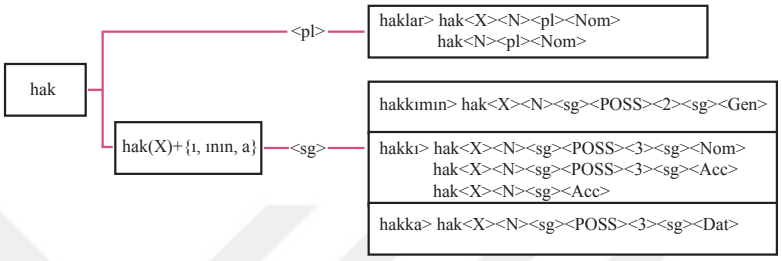


Figure 3.4 Visualization of lexicon entry *hak* in singular and plural forms with its trigger function X.

Epenthesis

In some cases, a phoneme is added between two morphemes in order to make the pronunciation easier (epenthesis). This process is only partly predictable and is therefore marked in the lexicon with a trigger symbol [71]. If a noun ends with a vowel, for example *silgi* ('eraser'), an /n/ is added before the genitive suffix -Yn. In other contexts, for nouns ending with a consonant, e.g., *kalem* ('pencil'), a /y/ is added.

The Aorist Suffix

A further verbal suffix in Turkish is the aorist suffix -Yr. It behaves irregularly with respect to vowel harmony. Four cases can be distinguished. All polysyllabic verb stems and the following 12 monosyllabic verb stems fall into the first category: *al* ('take'), *ol* ('become'), *öl* ('die'), *gel* ('come'), *kal* ('remain'), *bul* ('find'), *var* ('have'), *ver* ('give'), *vur* ('beat'), *gör* ('see'), *san* ('suppose'), and *dur* ('stop'). This list of verbs was taken from the work of Oztaner [70]. These verbs follow

the normal vowel harmony rules. The second category comprises all monosyllabic verb stems ending in a consonant, except those listed for the first category. If the vowel of the stem is a front vowel (see Table 3.8), the aorist suffix becomes -er; otherwise it becomes -ar. The third category comprises verb stems that end with a vowel. In this case, the vowel of the aorist suffix is deleted. The final case concerns negated word forms. After the negation suffix -m‹K›, the aorist is always realized as /z/.

Passive

The Turkish passive suffix can realized in three ways depending on the context. If the stem ends with a vowel, it is realized as -n as in oku+n *okun* ('to be read'), ara+n *aran* ('to be searched/looked for'), or koru+n *korun* ('to be protected'). If the stem ends with /l/, it is realized as -Yn, e.g., al+Yn *alın* ('to be taken'), gül+Yn *gülün* ('to be laughed'), or sil+Yn *silin* ('to be wiped'), and in all other cases as -Yl. Some examples are yaz + Yl *yazıl* ('to be written'), gör + Yl *görül* ('to be seen'), and soy + Yl *soyul* ('to be robbed'). For the verb çağır+Yl *çağrıl* ('to be called'), and so on, we use the trigger ‹del› for vowel deletion for passive word forms.

3.4 Design of the Gold Standard

Morphological analyzers analyze morphemes. This means that only the grammatical categories are included, which are realized as morphemes. The gold standard of morpheme analyses includes the correct grammatical morpheme analyses, which were used as a reference in the evaluation [72]. A gold standard (i.e., data that are manually annotated with the correct analyses) is an important part of any quantifiable evaluation. For our evaluation, we needed a list of word forms annotated with their correct morphological analyses. This raises the question of what the correct analysis of a complex word (homographs) such as *gözleme* (1. observation and 2. pancake) should be. We could analyze the whole word as a

lexicalized form, or we could split it into smaller parts according to word formation rules [73]. TRMOR generates the following analyses for the word *gözleme*:

göz‹N›l‹K›m‹K›‹N›‹SUFF›‹sg›‹Nom›
gözle‹V›m‹K›‹V›‹SUFF›‹neg›‹imp›‹2›‹sg›
gözle‹V›m‹K›‹V›‹SUFF›‹neg›‹imp››2›‹sg›
gözle‹V›m‹‹N›‹SUFF›‹neg›‹POSS››3›‹sg›‹Dat›
gözle‹V›‹N›‹SUFF›‹POSS›‹3›‹sg›‹Dat›
gözle‹V›‹N›‹SUFF›‹sg›‹Dat›
gözlem‹N›‹sg›‹Dat›
gözlem‹N›‹sg›‹POSS›‹3›‹sg›‹Dat›
gözleme‹N›‹sg›‹Nom›

TRMOR provides analyses of varying granularity for the entry *gözleme* in four categories, with the granularity of the analyses given in decreasing degrees, from the fine-grained analysis to the near representatives achieved. From *göz* ('eye'), a new semantically not-very-wrong word arises through the derivational suffix -lKmK. We decided to choose the right level of granularity if the analyses were close to the meaning. This means that if the word *hazır* ('ready'), for example, is analyzed in a fine-grained manner as *haz* ('enjoyment') + the transitive suffix -Yr, we have a wrong analysis. During this process, an interesting question arises of which granularity of the analysis should be obtained [51].

3.5 Evaluation

We evaluated TRMOR based on two large corpora and compared it to the TRmorph analyzer for Turkish in order to assess the quality and coverage of the system. During the development of TRMOR, we initially tested the rules with example words. Then a word list was extracted from an online newspaper [2] and non-words

[2]http://www.haber7.com/genel-saglik/haber/732846-cocuklari-istismardan-nasil-koruyabiliriz

and numbers were removed from the document. This evaluation raised the question of whether it is more important to increase coverage or to avoid over-generation, since high coverage usually leads to more over-generation.

The two morphological analyzers used in the present work include the state-of-the-art rule-based system TRMOR, which has a 94.12% success rate in morphological analysis of Turkish words. One thousand words chosen at random from Wikipedia word lists in Turkish were used for the word lists. The well-known TRmorph, on the other hand, utilized Wikipedia with about 29 million tokens and achieved 85% success with the analyzed words.

Table 3.10 illustrates the frequency of 1 000 words from Wikipedia word lists taken from [63], randomly extracted. The column "words in range" represents the number of words selected from the corresponding frequency range of 1 000 from the Wikipedia word lists. For example, the first entry in Table 3.10 indicates that 159 words of the word lists are selected from 16-20 in terms of the frequency range.

Table 3.10 Frequency distribution of Wikipedia word lists for 1 000 words.

Frequency range	# words in range
16-20	159
21-65	476
66-100	103
101-25,801	262

For the evaluation, a word corpus of 1 600 000 words was used with its popularity, which was extracted from books. From the corpus, the gold standard itself is produced within this scope semi-automatically. From 1 600 000 entries, 100 000 sentences will be randomly selected in each case. Of these entries, some represent the right analysis, some were correct but did not produce all the right analyses, some were false, and some had no analysis, meaning that they were not analyzed

by TRMOR because either the entry in the TRMOR lexicon was not present or was not acted upon. These are non-words. For all of these 1 000 000 sentences, the gold standard is executed. In the case of missing analyses, the possible analyses were completed, and the false analyses were corrected and also completed. In the performance of the gold standard, only the entries that have been spelled incorrectly will be corrected using the ANN-based model. The evaluation will then be run on this file. In the second phase, unclear analyses collected in the first step are automatically deleted. All other analyses are then examined in detail.

A corpus with frequency information was used for the evaluation, which was extracted from the Vikipedi (the Turkish version of Wikipedia) lexicon, and a word list of the word types was extracted. Frequencies of 1 were deleted from the corpus, and the evaluation was carried out afterwards. The gold standard on this body itself was produced semi-automatically. From 489,274 entries, which resulted after the deletion of frequency 1, 1 000 entries determined at random from the 100th entry were selected. From these entries, some were the correct analyses, some were correct but did not produce all the correct analyses, some were wrong, and some had no analysis at all or they could not be analyzed by TRMOR because the entry was not available in the TRMOR lexicon or it was a non-word, or else the word was wrong orthographically or grammatically in the Vikipedi corpus. All of these 1 000 words were executed on the gold standard. In the absence of analyses, the possible analyses were completed and the wrong ones were corrected and completed. During the execution of the gold standard, we handled not only the entries that were incorrectly written orthographically but also those grammatically incorrect. The evaluation was then executed on this file.

In the second phase, unclear analyses were automatically deleted. All other analyses were then examined in detail. When processing 1 000 word forms using TRMOR, 1 343 analyses were obtained.

Table 3.11 shows the calculation of precision and recall in order to evaluate all the expected analyses for each word form and thus each output comparing the output of the tool to one of four categories: true positive (candidate suitable and correct analysis), false positive (candidate suitable and wrong analysis or candidate not suitable and the tool does not provide any analysis), true negative (candidate not suitable and tool does not provide analysis), and false negative (candidate is suitable but analysis/analyses is/are not provided).

Table 3.11 Wikipedia test corpus evaluation results.

	# of words	Precision (%)	Recall (%)	F-Measure (%)
Wikipedia	1 000	94.12	79.80	86.37

3.6 Summary

We have presented the TRMOR system, a finite-state morphological analyzer for Turkish that covers Turkish inflection, derivation, and some composition. The treatment of composition is interesting and difficult, since almost every declination of compound words differs from others. For the word *acemborusu* (the Turkish common name of 'Bignonia radicans'), the system fails for the compound marker '*su*', except for the genitive marker for the nominative case, whereas for the word *ayçiçeği* ('sunflower'), 'i' is the compound marker. It seems that the compound markers occur differently according to the second noun, as in the first example for *boru* ('tube') and in the second example for *çiçek* ('flower').

The system analyzes the word forms, decomposes them into morphemes, and tests them for accuracy and correctness. Its accuracy and coverage were tested on real data (an online newspaper) and were compared to those of an existing analyzer. Besides having no cost, it is precise. However, the quantitative results of a system mean little if morphologically incorrect analyses are also accepted. TRMOR can be freely used and modified for various tasks including machine translation,

named entity recognition, semantic analysis, or OCR error detection, and it can serve as a basis for the creation of morphological word-form analyzers for related languages.

CHAPTER 4

ARTIFICIAL NEURAL NETWORK-BASED LANGUAGE MODELING: LSTM NETWORKS

Language models have critical importance in the pre- and post-processing of OCR. For long time intervals of sequences, the LSTM fulfills the requirements because the LSTM is capable of solving problems with long-term dependencies.

Structure. This chapter is structured as follows: We introduce some theoretical and technical background in Section 4.1. Based on that, we present a detailed description of the LSTM model in Section 4.2, including an example of performance calculation on how it performs and furthermore the preparation of data for the model before we conclude the chapter with a summary in Section 4.3.

4.1 Introduction

Artificial neural networks (ANN) are a simulation of biological neural networks. ANNs combine artificial neurons in order to process information [74]. The main types of neural networks are feed-forward neural networks, which are the most common type of neural network in practical applications. Feed-forward neural networks cannot remember the context to generate the output. Also, they cannot handle long sequences of input lengths.

The other neural network architecture is the convolutional neural network (CNN), which achieves good results in speech recognition in NLP. The CNN can have

local information from data, but it is not possible to learn the long sequential dependencies. On the other hand, the recurrent neural network (RNN) is special for sequential data [75]. The RNN can learn and is able to handle small dependencies up to tree-gram dependencies. For long sequences, however, RNNs are unable to learn long dependencies.

RNNs are able to recognize sequences and work with dependencies, but unfortunately this ability is limited. If there is only a small time gap between interdependent data, the RNN will be able to recognize this relationship and draw the right conclusions. However, if the time interval between input of the data and the time at which they are needed for a result becomes very large, the RNN can no longer establish this relationship it means that one can build a language model that predicts the next word depending on the previous one. The LSTM is an extension of the RNN, which is a type of ANN, and we used LSTM for this model because it is able to master the long-term context dependency.

4.2 Related Works

RNNs are widely used in sequence modeling tasks, such as speech recognition [76], and also [77] for LSTM-based RNNs, in speech recognition [78], handwriting recognition [79], and optical character recognition [80]. Neural network models play an essential role in state-of-the-art NLP tasks, such as error detection and correction of OCRd text documents [81], neural text classification [82], named entity recognition [83], part-of-speech tagging [27], semantic parsing [84] and question answering [85], language generation [86], multi-document summarization [87], spell checking [88], machine translation [26], and image generation [89].

The LSTM model is based on OCRd data that provide better performance during correction of the errors without using a language model. LSTM models show

better results when used for language-independent OCR [41] since, for example, our LSTM work is close to the model of D'hont et al. [23].

Islam and Inkpen showed that statistical n-gram language models are effective to solve real-word errors [90]. However, there are some weaknesses of n-gram language models, one of which is that the number of possible n-grams increases exponentially with the length of the context [7]. The idea of using neural network-based language models is based on this observation, and it tries to overcome the exponential increase of parameters by sharing parameters among similar events [7].

Several approaches have been proposed for correcting OCR errors. Some of the methods are dictionary-based, while some are state-of-the-art neural network-based, especially among LSTM approaches. In order to have a model using LSTM, a small corpus is used to make corrections on a character level. However, whenever we have more training data, better results can be obtained because it learns the relations between characters to form a word [10]. In order to train character-level LSTM, large data are used to learn to generate text like it one character at a time [54].

Language models are an essential step in OCR. LSTMs have been used in a wide range of problems including text recognition in images and generating language models. Bidirectional LSTM models are based on the idea that the output at a particular time may not only be dependent on the previous elements in the sequence, but also on the succeeding sequence. An LSTM learns the error model from the erroneous words during training [91].

The LSTM models based on OCR provide better performance in correcting errors even without language modeling. LSTM models show good promise to be used for language-independent OCR. For correcting the OCR errors, there are many LSTM methods. These include character-based and word-based approaches, but there is a difference between them in the architecture and in the input and output. We need a

relatively small corpus to train the correction model while the LSTM is learning the correction on a character level. However, the more the training data, the better the results as it learns the relations between characters to form words [92].

The word-level implementation of LSTM results is performed through a bag-of-words representation. In contrast, our implementation is at character level. We have a list of all possible characters of Turkish and others through text. This means that instead of keeping a list of words, we keep a list of countable different characters.

Previously, the artificial neural network was not used in character-based OCR error correction tasks. The one indication is the excellent performance of the LSTM-based approach. It is a simple approach to implement and needs less time and effort to train after hyperparameter tuning. The prediction using LSTM is fast. However, other approaches are time-consuming, are labor-intensive, and require effort to construct and combine different components.

4.3 LSTM

LSTM [93] is designed to address the limitations of the vanilla RNN [94]. LSTM is one of the most widely used recurrent structures in sequence modeling. Its goal is to use gates to control the information flow in the recurrent computations, although its practical implementation based on soft gates only partially achieves this goal and it is easy to experience overfitting [95].

LSTM networks are a special type of recurrent network that can work with long-term dependencies. They are designed to specifically solve this problem, because storing information over a long period of time is their default behavior and not something that has to be learned with difficulty. They consist of memory cells into which information can be written and read out again.

BiLSTM model: A bidirectional LSTM [96] is typically used to deal with both left and right context. The bidirectional LSTM consists of two unidirectional LSTMs, one encoding the inputs and the other decoding. The output of bidirectional LSTM is then a sum of forward and backward RNN outputs. In this way, bidirectional LSTM is processing information from both the preceding and the following context [38].

Our LSTM model, which is based on the model developed by D'hondt et. al. (2017) [23], uses BiLSTM to convert the input sequence to a one-hot vector while encoding the characters, and then another deep BiLSTM decodes the output sequence from the vector. The BiLSTM consists of two identical hidden layers. One layer scans the input in the forward direction, while the other scans the input in the backward direction. This is done to access the context from previous and future time steps.

LSTM models are applied for a wide range of tasks, from the recognition of text in images to model generation for language. The most popular of the LSTM models are the BiLSTM models, established on the premise that output at a certain moment may be dependent not only on the prior elements from the sequence but on the succeeding sequence, as well. BiLSTM learns error modeling from erroneous words that it encounters during training.

The BiLSTM model that was introduced by Graves et al. [97] consists of two unidirectional LSTMs, one encoding the inputs and the other decoding. The output of the BiLSTM is then a sum of forward and backward RNN outputs. In this way, BiLSTM processes information from both the preceding and the following context [98]. The model allows an arbitrary number of layers. Our model has only one layer, because that is suitable for our model producing good performance. The softmax activation function is utilized for predicting the probability of every character present in the output sequence over the output alphabet at each time step.

The ADAM optimizer is used because it gives better results for many models [99]. The learning rate value is chosen as 0.001. The numbers of epochs used in our experiments are 50 and 100. We train the model for k-fold cross-validation of only 50 epochs.

4.4 Hyperparameters of the LSTM Model

The hyperparameters of the model are the number of epochs, batch size and sequence length, and step size, all fixed according to the data type. The tuning of the hyperparameters should be specially performed for each data type. Thus, the hyperparameters were tuned experimentally for good performance and were set as follows: sequence_length = 30, number_of_nodes = 512, step = 3, num_layers = 1, batch_size = 64. The best and standard dropout score is 0.5.

The most critical of the parameters is the size of the hidden layer, representing the number of LSTM cells, or the number of nodes present in the hidden layer. Increasing the number of cells does not necessarily increase the performance. However, it increases the training time. We do not need to use more layers, because the program yields good results for one layer. If the program does not yield good results, we can perform more tests to increase the number of nodes, batch size, or dropout score. Hyperparameter tuning was a very time-consuming part of this research.

The LSTM model was trained repeatedly on CPU and GPU hardware with different sizes of data and the suitable values of the hyperparameters for the model were empirically identified. We created a table for finding the best parameters of the LSTM model and trained the model with two data sets, one a large data set containing about 65 000 words and the other with 1 031 words. The large data set was run on a GPU server, which resulted in good performance relative to the small data set.

The input for this model is a corrupted text file as a parallel corpus of OCR output and its corresponding clean text file. Before we can feed input into the model for training, we must initially perform pre-processing, and we must get our training data from corrupted strings [23].

A total of 1 031 words of OCR data including only 8 OCR erroneous data were fed into TRMOR. TRMOR produced 3 wrong results from 8 OCR erroneous data and their type was real-word error. Also, it did not analyze one of them because that data was not in the lexicon. However, if it were to be analyzed, it would produce the wrong result because it is a real-word OCR data.

Moreover, we train the test data in the LSTM model. One of the erroneous words out of the 8 erroneous OCR words was not corrected in all tests. Other reclamation statistics are shown below in tables. We conclude that if we use both LSTM and the rule-based system together, as shown below in the results from 10 experiments, we have better performance results.

4.5 Data

The Turkish language has some special encoding because of its special characters. Finding the encoding of the text for the LSTM model was very laborious. The encoding type that we selected for the input data set was cp1254. In this work, we used two types of corpora, namely one corpus for representing the clean text data for training and the other a real-life OCR corpus for testing. We used the clean text corpus to gain artificially corrupted strings. These corrupted strings were then used while training the model to learn.

Preparation of the Data

Our model requires clean data for training from the OCRd corpus to learn a character-based statistical language model using BiLSTM.

Sequential data pose a challenge for neural network architectures because they require that the dimension of the inputs and outputs be known and fixed [4]. In this dissertation, we use a straightforward application of the ANN-based LSTM architecture that can solve general sequence to sequence problems. The LSTM's ability to successfully learn on data with long-range temporal dependencies makes it a natural choice for this application due to the considerable time lag between the inputs and their corresponding outputs [100]. There have been a number of related attempts to address the general sequence to sequence learning problem with neural networks.

We used two different sets of OCR data, one large corpus of data for the GPU testing and one small set of data, for our proposed work. The first part of the data includes approximately 5 000 words and training of these data takes 100 epochs. The total calculation time of these data is 16 hours for 100 epochs, and one epoch takes 9.6 minutes on GPU. The second part of the data, which has 1 031 words, is extracted from the existing large data set. The latter is also used in the morphological analyzer in order to detect OCR errors having rule-based systems. The novel hybrid model that combines the rule-based XMOR and ANN-based systems has better results. The clean data from test data sets for each part (large/small test data) are required for training the ANN-based model and the test data set of the OCR corpus data is used for testing the ANN-based model.

The words in both the training and test data must exactly match, but there are errors so that the LSTM model learns. We used the clean text data to gain artificially corrupted strings. These corrupted strings were used for training the model. The clean corpus data must be prepared at the beginning in order to do pre-processing of the training data.

After OCRing the text, we saved them in a plain text format. There are some special characters in the Turkish language, so we changed the collected data using Turkish

encoding. The encoding versions of Turkish are ISO 8859-3, ISO 8859-9, OEM 857, and Windows-1254. Since we worked on the model using Linux and Windows machines, we used the encoding UTF-8 for Linux and ISO 8859-9 for Windows, respectively.

There are two data sets in this study: training and testing data. The training data set is labeled as clean, meaning that there are no errors in this data set. Moreover, we create noise data during the training phase to increase the error correction rates. In other words, we apply a corrupted string to our model so that our model can correct the corrupted words during the testing phase. The user can also change the level of corruption in order to improve the model.

We use three types of data sets, namely test, test_target, and train. The training data set is used for training the model. After training the model, we evaluate the trained model with the test data set. The test data set includes noise or unclear data. Then the trained model generates a text as the output, and we must evaluate the model to determine whether the results are credible. We have test data from OCRd text for testing and training data for training without erroneous words.

4.6 Components of the LSTM Model

Artificial neural network-based systems can generally be divided into the three parts of training, tuning, and evaluation, each of which we present in detail along with illustrative examples. The model inputs are characters with indexes. These characters are embedded, i.e., each character in the input sentence is represented by a one-hot row vector. The model allows an arbitrary number of layers. In our model, we used one layer that gives good performance. The softmax function is used to produce the probability distribution over the output alphabet at each time step. For the one-layer BiLSTM encoder/decoder, the size of hidden states is 1 024, the word embedding size is 1 024, and dropout percentage is 0.5. As an optimizer,

the ADAM optimizer is used since it gives better results for many models [99]. The learning rate here is the default number 0.001.

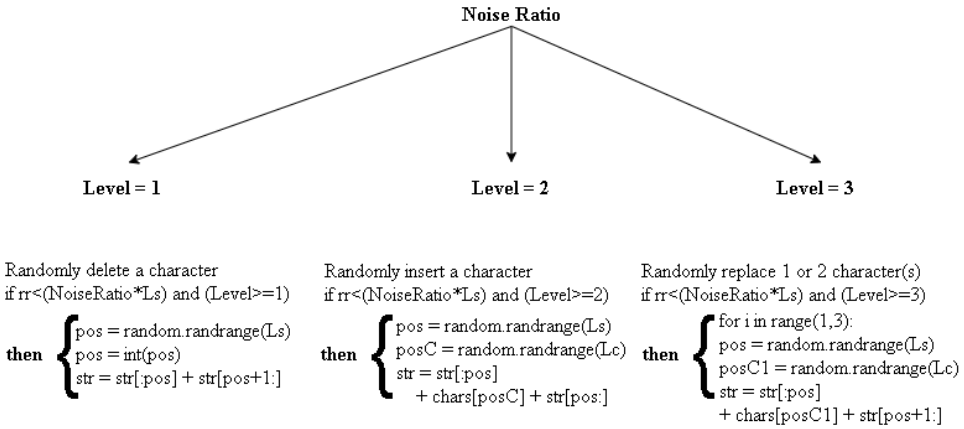


Figure 4.1 Illustration of corrupting string function.

Figure 4.1 illustrates the noise ratio definition of the corrupting string function. The noise ratio compares the level of a desired signal to the level of background noise. There are three types of noise. Level 1 is for deleting a character randomly. In level 2, the noises of level 1 and level 2 are randomly applied, and in level 3, all noises are randomly applied. In this way we gain our training data. The noisy data are created from relatively clean data to train the model. Table 4.1 below describes with an example string sequence how the noise ratio impacts the output string sequences.

Users can choose the noise ratio of the model as shown in Figure 4.1, and different noise ratio examples are given for corrupted strings depicted in Table 4.1.

4.6.1 *Performance Measure of the LSTM Model*

The ANN-based LSTM model was run on two different data sets as mentioned in previous chapters. We want to calculate the performance of the model for each input string, instead of for each character. In order to check how our model learns

Table 4.1 Examples of corrupted strings w.r.t. the noise ratio on the clean corpus. Input string (36 characters length): dünürler eyden.nce (ve hâlâ) ilginç.

Noise ratio	Output string
0.001	dünüareer eyden ncel(v hâlâ) ilginç
0.003	dünorlereyden mrnd sve hâlâ) ilkinç
0.005	dünürlm eyden pnce(vc hâlâ) ilgoinşğ
0.01	dütnürler pyen nce (ve hâyâ) ilginç
0.02	djünerlrbykmderscgüşıvüıshâr lgisçv
0.03	drşsrleç ikbytdj ıht (noe ââuznlgive
0.04	diğüfeueeydruzcpcecu (mö üâçıgctlfnç

in the training phase and how the data type and amount of the training data affect the performance of the model, the calculation of the performance of the model is done as in the following scenario. Assume that we have the following input text: ***'dâhileri değil'***. When we apply the input text (a) to the LSTM model, it will generate a performance output. For example, it generates the following text (corrupted characters are highlighted in red font):

Step 1: In order to calculate the performance of this input string (a), we need to represent the characters as integers. In this step, all characters of ground truth strings and the corrupted string (c) are represented as integers in an array.

Input string (a) = 'dâhileri değil'

Corrupted string (c) = 'dâlileri değı1'

d	â	h	i	l	e	r	i		d	e	ğ	i	l
23	12	22	16	11	9	13	16	8	23	9	14	16	11
d	â	l	i	l	e	r	i		d	e	ğ	i	l
23	12	11	16	11	9	13	16	8	23	9	14	16	7

a	23	12	22	16	11	9	13	16	8	23	9	14	16	11

c	23	12	11	16	11	9	13	16	8	23	9	14	16	7

Step 2: In this step, we differentiate the representation of character indexes of input string and corrupted string for each character. After that, according to the results,

the places in the array with zero values show that a_i and c_i have the same values, as shown below in an array.

d = input string array after differentiation

```
for (int i = 0; i<sizeof(d); i++)
```

d_i = |a$_i$ - c_i|

d	0	0	11	0	0	0	0	0	0	0	0	0	0	4

Step 3: Through flipping and binarization, we set each zero value. This means that the character representation is true to one and the non-zero value to zero.

```
x = binary array for input string
for ( int i = 0; i<sizeof(d); i++)
if (d_i == 0)
    x_i = 1;
else
    x_i = 0;
```

x	1	1	0	1	1	1	1	1	1	1	1	1	1	0

The performance of the model denoted as ***P*** for the input string *'dâhileri değil'* is calculated and found to be 85%.

$$\boldsymbol{P} = 100\ (x_i)sizeof(d)$$

$$= 85.71\%$$

4.7 Summary

A new technique based on LSTM recurrent networks is implemented to resolve the erroneous words of OCRd text documents for Turkish. An introduction to LSTM was provided in Section 4.1. For training, the OCRd word forms are used with their corresponding ground-truth word forms. In Section 4.2, some information about LSTM was given. We described the performance measure of the LSTM model via an example calculation in Section 4.4.1. In this chapter, we have proposed a novel zero-annotated data approach to OCR post-correction. For our model, we used BiLSTM to build an encoder-decoder model.

CHAPTER 5

EXPERIMENTAL ANALYSIS

5.1 Introduction

Natural language is an intricate object for computers to handle. Philosophical debates aside, the field of NLP has witnessed a paradigm shift from rule-based methods to statistical approaches, which have been dominant since the 1990s [101]. However, the accumulation of data and computation resources that have recently been developed, in addition to new algorithmic techniques, have allowed the domination of this section of machine learning in numerous areas of artificial intelligence. In the beginning, in perception tasks, such as computer vision and speech recognition, and gradually, in NLP, since about 2013 [101].

We conducted our experiment using both rule-based XMOR and ANN-based implementations as contrasting methods.

Structure: The organization of this chapter is as follows. The accuracy of each approach, based on two datasets, will be discussed. The results clearly showed that the linguistically-motivated morphological analyzers presented in previous chapters performed better than the artificial neural network approaches that were presented in Chapter 4. In Chapter 3, confirmation of the improvement in the rule-based motivated approach compared to the neural network-based approaches (in Chapter 4) was exhibited in a combined hybrid system.

The structure of the rest of this chapter is as follows: A brief review is presented of the different approaches and a comparison and discussion of the results of evaluations for rule-based XMOR and ANN-based approaches have been given in Section 5.1. In Section 5.2, among others, the context-dependent error model for real-word errors of OCR is described. Section 5.3 contains the experimental results. Finally, Section 5.4 presents a summary of the chapter.

5.2 Testing the Rule-based and Artificial Neural Network Approaches

We tested rule-based approaches for Turkish via TRMOR and TRmorph and compared these to get the input data for the neural network-based approach, namely the ANN-based LSTM model. The experiments were divided into three groups for the hybrid system. The reference data set was fed into the rule-based XMOR component to ensure that the output would represent annotated data sets.

The accuracy of the rule-based approaches was calculated for TRMOR and TRmorph separately using two completely different data sets, while 10-fold cross-validation was chosen for the neural network-based approach to get the approximate average model accuracy. For the evaluation of the model, we used k-fold cross-validation, with results given in tables below.

The experimental results obtained here reveal that adaptation of the hybrid approach generates the best accuracy. The evaluation of the OCRd data and social media data of the model is illustrated in more detail below. Moreover, we trained the testing data in the ANN-based LSTM model. One of the 8 erroneous OCR words was not corrected in any tests. Other reclamation statistics are shown below in the tables. We conclude that if we use both ANN-based LSTM and rule-based systems together, as the results show below for 10-fold experiments, we will have better accuracy results. Table 5.1 illustrates how many times the model responded to the

errors in the testing data in every test phase. The overall analysis was based on 8 OCR errors for 10-fold testing of the model.

Table 5.1 Test average for 8 errors of OCR data using ANN-based module in 1b in the algorithm.

	Corrected words	Non-corrected words
Average (%)	62.62	67.30

Using OCR testing data, 1 031 tokens were fed into the morphological analyzers of XMOR (TRMOR and TRmorph), including 8 erroneous strings. TRMOR produced 3 wrong results from those 8 OCR erroneous strings, which were real-word errors. Also, it did not analyze one of them because that data was not in the lexicon. However, if it were analyzed, it would produce the wrong result because it is a real-word OCR error.

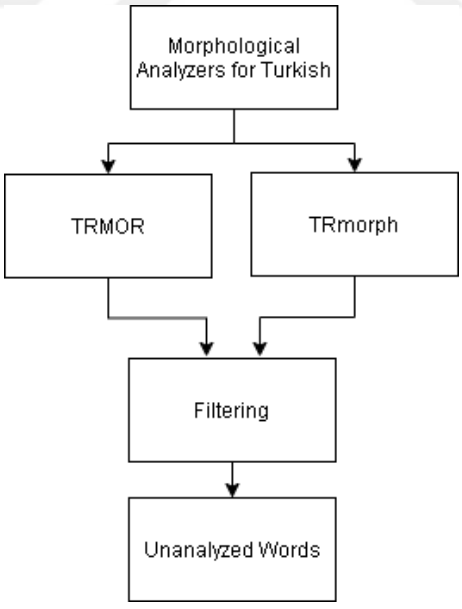


Figure 5.1 Filtering the unanalyzed words from morphological analyzers.

Model responses to strings are shown in Table 5.2. In this table, the non-word error ***anzalar*** could not be corrected in either system. In any test cases we assume that the

location of this word is at the very beginning of the data. The string ***çağrıştırdığım*** as a real-word error was also not corrected in any of the tests. Solving this type of error is a challenging problem for all NLP language modeling. Table 5.2 shows typical OCR errors such as /r/ + /ı/ => /n/ in strings s1, s2, s3, and s6 that are corrected at higher percentages.

Table 5.2 Distributions of erroneous words in total test cases for OCR data.

	s1	s2	s3	s4	s5	s6	s7	s8
String	konulan	kitaplan	onlan	çağrıştırdığım	farkmdalığı	aynca	lhâlde	anzalar
Ground truth	konuları	kitapları	onları	çağrıştırdığını	farkındalığı	ayrıca	hâlde	arızalar
Corrected words (%)	60	70	60	-	60	70	70	-
Non-corrected words (%)	40	30	40	100	40	30	30	100

Turkish social media data were used as a second set of testing data for comparing the accuracy of the model. The 10-fold test method was used to obtain average model accuracy, and 1 142 words from social media data were used for testing and training. There were 327 words in TRMOR and 75 words in TRmorph generated as words with no results from this data set. Seventy-one of these words did not have results generated in either system. In Table 5.3, we depict the distribution of unanalyzed words from both analyzers for the social media data. Figure 5.1 illustrates the extracting of unanalyzed words from the morphological analyzers.

Evaluation results for 10 tests of the LSTM model result in 74% on GPU and 38% on CPU on average. These results show that the large data set, trained on GPU, has better correctness results compared to the small data set on CPU.

Table 5.2 above describes how many times the model responds with errors of testing data in every test phase. Overall analysis is based on 8 OCR errors for 10 tests of the model. Responses of the model to strings are listed above in Table 5.2. As shown in the table, the erroneous word of the non-word type, ***'anzalar'***, could not be corrected in both systems in any test cases. We assume that the location of this word is at the very beginning of the data. The string ***çağrıştırdığım*** for the error

type of real-word error is also not corrected in any of the tests. The solution of this case of error is a challenging problem in all NLP language modeling.

Table 5.3 TRMOR and TRmorph analysis for social media data.

	TRMOR	TRmorph	Both (common words)
No result (%)	28.60	6.60	6.22

The words without generated results were fed into the LSTM model to see how the model is handled.

The LSTM model was executed for the two different data types. As shown in Table 5.4, social media data normalization gave better results on average than the OCR data for 10-fold testing.

Table 5.4 Comparison of accuracy for different data types using the LSTM model.

Performance	OCR	Social media
Maximum (%)	98.13	97.18
Minimum (%)	10.93	11.27
Average (%)	29.62	57.88

There are 158 words in TRMOR and 30 words in TRmorph that generated no results from among 1 031 words. Table 5.5 illustrates the performance analysis of the models (TRMOR, TRmorph, and LSTM) for OCR data and social media data.

Table 5.5 Distribution of accuracy analysis of models.

	1. TRMOR	2. TRmorph	Hybrid 1 and 2 with LSTM (average)	Hybrid 1 and 2 with LSTM (maximum)
Social media (%)	72.24	93.43	97.38	99.82
OCR (%)	84.67	97.09	99.03	99.90

Table 5.5 indicates that the current hybrid method combines the rule-based and the long short-term memory-based techniques to improve the morphological analysis accuracy for OCRd data to 99.03% and social media data to 97.38% on average, which constitutes state-of-the-art morphological analysis for Turkish to the best of the author's knowledge. Moreover, the maximum accuracy is 99.90% for OCRd data and 99.82% for social media data. We observed that the accuracy was increased by 0.97% for OCRd data and by 3.6% for social media data when we used a maximum of k-fold corrected values, which indicates that LSTM has a better correction rate for using social media data compared to OCRd data.

- The current hybrid method combines rule-based XMOR and ANN-based techniques to improve the morphological analysis accuracy for OCRd data by 0.97% on average and for social media data by 3.6% on average for Turkish.
- The current hybrid method combines rule-based XMOR and ANN-based techniques to improve the morphological analysis accuracy for OCRd data to 99.9% using the maximum of performance and for social media data to 99.82% on average using the maximum of accuracy for Turkish.
- The current hybrid method combines rule-based XMOR and ANN-based techniques to improve the morphological analysis accuracy for OCRd data to 99.03% and for social media data to 97.38% on average for Turkish.

5.3 Discussions of the Results

The hybrid model implemented herein was compared with a model combining the neural network-based and rule-based approaches. The combined model exhibited higher performance than the single-rule-based XMOR mechanism. The ANN-based LSTM system, which was trained on erroneous ground truth data, was able to correct many of the errors [4].

Ground truth samples of the two approaches were presented in Table 3.1 of Chapter 3. The results showed that, for OCRd data, the normalization approach more effectively corrected the errors as a result of the OCRd data having more dependence on contextualized text than social media data when solving real-word errors. There were some repetitive errors in the social media data presented in Table 3.1, such as '*şişti*i' for which the model provided better results, while correcting abbreviations like '*tşk-teşekkür*' required more edit operations [102].

One of the methods used embedded pre-trained neural words as a means of including contextual information for learning the canonical form of a selected noisy word. The results showed that the approach proposed herein performed better than the other models, even though the proposed model was completely unsupervised. The latter method comprised an encoder-decoder model and BiLSTM architecture. It did not utilize contextual information, rather, it learned the error patterns through the use of a great deal of noisy-canonical word pairs [102].

Previous chapters dealt with detailed information regarding the hybrid model modules. This chapter presented results that showed that the LSTM was able to yield significantly better normalization results for the OCRd data than for that of social media. Several indications presented that the LSTM-based approach was able to generalize much better with the unseen samples than the rule-based approaches that were applied to the OCR error-correction system.

It is a simple approach that requires less effort and time to train. Using LSTM, prediction can be performed quickly, while other the approaches are labor-intensive and time-consuming, and effort is required to construct and combine the different components.

The rule-based approach may be incapable of handling all of the spelling variations when the OCR encounters character misrecognition. Misrecognition results in the OCRd text being mismatched with a real word. Some of the approaches fail to

change unseen samples when they have not been included their models. In the word list approach, unseen samples are unable to be processed. However, with the implemented LSTM, all of the tokens in the test sets were processed, and the model exhibited a significant ability to very quickly predict all of the spelling variations [74].

The error model was designed to function like a filter to handle mistakes in the recognition outputs [74]. Finite-state-based morphological analysis is the most traditional and simplest model used to make this kind of a recognition filter. Correcting OCR errors via the LSTM model normally comprises the generation of a list of word forms that belong to the selected language. The benefits of empirical and rule-based approaches in morphological analyzers can also be incorporated in hybrid systems. Generally, there are different methods for achieving this. As an example, they can be run at the same time and a post-processing step can be used to combine their outputs, or linguistic rules can be assigned as two-level phonological rules via frequency statistics [57].

Rule-based morphological analyzers possess sets of rules that have been previously well-defined and comparable finite-state machines for employing the rules. Improvements in rule-based models vary depending on the language that is modeled. Commonly, to improve rule-based models, the network model is enlarged, the rule-set and lexicons are enriched, a register is employed, and most often, statistics are embedded into the models [103].

Rule-based morphological analyzers have some advantages that can be also be disadvantages. This is particularly related to components of the system that are language-specific, because usually, they cannot be applied for another language. As a result of this issue, unsupervised approaches were developed for morphological analysis [57]. Although it was stated that morphological analyzers were not as expensive to create as, for example, syntactic parsers, considerable effort is still

required, especially in relation to a high-coverage lexicon [57].

Over time, language changes, and although morphological analyzers are able to easily handle new words that their lexicon contains, they cannot analyze basic words that have not been included, such as new technical terms or proper nouns. Therefore, TRmorph and TRMOR have both been continually maintained and extended in recent years. Hence, not many morphological analyzers are freely accessible as a result of the expense caused by such maintenance [57]. Rule-based natural language generation systems [104] have been quite successful but they suffer from some limitations. They require extensive human effort.

We tested rule-based approaches for Turkish via TRMOR and TRmorph and compared these to get the input data for the neural network-based approach, namely the LSTM model. The experiments were divided into three groups for the hybrid system. The reference data set was fed into the rule-based component to ensure that the output would represent annotated data sets.

The evaluation results show that the current hybrid method can improve the morphological analysis performance to 99.90% for OCRd data (considering the top suggestions [105]) and to 99.82% for social media data using maximum performance.

An approach where one combines the artificial neural network-based model with a more knowledge-based system such as a finite-state-based morphological analyzer, such as we presented in Chapter 2, is probably a viable approach. On the other hand, a simple FST-based filter will detect errors. Some previous work, also on social media normalization, has made use of neural techniques. In the last few years there has been a lot of work focusing on social media [106].

5.4 Summary

In this chapter, we have shed some light on the experimental results of the hybrid model. We have discussed the results for two different data types. Along the way, we gave numerous illustrative examples. These were for both rule-based XMOR systems and the ANN-based system. Finally, this chapter presented some of the experimental background upon which the previous chapters about the hybrid model were built.

CHAPTER 6

CONCLUSIONS AND FUTURE WORKS

6.1 Introduction

In this chapter, we provide a summary of the main conclusions of the dissertation. This chapter also provides a list of research directions to lead subsequent research in the fields of semi-supervised hybrid models improving the XMOR mechanisms.

6.2 Conclusion

In this dissertation, we have presented LSTM, a language-independent character-based encoder-decoder architecture for correcting noisy data, and TRMOR, a new Turkish morphology data set manually confirmed to have 96% inter-annotator agreement [107]. Our error analysis indicates that morphological analyzers can cause a large amount of real-word OCR errors [108].

The aim of OCR post-processing is to both detect and correct errors in the input text. The major difficulty in this task is posed by ambiguous words. However, if we know the previous and subsequent words of the sentences for each of the long dependencies, it is also possible to disambiguate the ambiguous words using LSTM. In our task of error detection in OCRd document output, we use the LSTM model. Our model is closely related to the work of [23], which gives good results for a small data set. We run the model for a large data set, and this necessitates a long time for

training, so finding the best parameters of LSTM takes a long time to complete. For every type of data, one should train the model anew, and it takes a very long time to find the best parameters. Therefore, it is very laborious with slow computers. We want to compare our BiLSTM model for Turkish with the model for French. Our model is the first work in LSTM modeling for Turkish.

Several studies have been conducted on the correction of OCR errors. OCR word correction using a noisy channel model is applied in [109]. Some of the applied methods use dictionary-based approaches while some use state-of-the-art neural network-based LSTM approaches. To obtain a model using LSTM, a small corpus is used to perform corrections at the character level. However, better results can be obtained with more training data because the model will learn relations between characters and accordingly be able to form words [92]. To train character-level LSTM, large amounts of data are used to learn how to generate text character by character [54]. However, it is also argued that higher volumes of training data do not necessarily lead to significant improvement [1], like in our case.

This study combines the rule-based morphological analyzer TRMOR with a neural network-based LSTM model for more precise comparison of rule-based and neural network-based systems. First, the LSTM model corrects the word errors in corpora obtained from OCRd documents, and later the success of the model is proved with social media data obtained from Istanbul Technical University [52], [53]. We explain the morphological analyzers TRMOR [55] and TRmorph [63] and the LSTM model in detail in Sections 3 and 4. We investigate the results through a comparison of the morphological analyzers and LSTM results and we run the LSTM model iteratively to improve the performance. In summary, our research accomplishes the following:

- Performances of TRMOR and another state-of-the-art analyzer, TRmorph, are investigated.

- Improvements obtained for TRMOR and TRmorph using the neural network-based language modeling approach LSTM are observed.

This dissertation presents results from a morphological analysis of the Turkish language via a rule-based approach with a finite-state mechanism and a neural network model based on LSTM architecture. We have proposed a hybrid approach to combine the principles and techniques of hand-crafted rule-based and state-of-the-art statistical or ANN-based approaches to bypass the specific problems of traditional and state-of-the-art ANN-based methods and simultaneously preserve their advantages.

6.3 Future Works

We will build a system to find out the content of texts using these large Turkish corpora. In order to disambiguate ambiguously occurring words, we apply a language model based on neural networks, which is a way to learn more sophisticated functions to improve the accuracy of our probability. Ambiguity resolution for words in sentences is a great challenge in natural language processing. It is still more difficult for agglutinative languages such as Turkish, Finnish, Hungarian, etc. Since Turkish is both an agglutinative and an economical language, sentences can be expressed in a few words, and most words can have more than one meaning, as can sentences.

There are two types of ambiguities. First, the word alone may have many meanings, and this leads to alternatively interpreting the sentence. Second, the suffixes may cause the case markers of a word to be ambiguously interpreted. An example of the first case is as follows: ‘yüzdün yüzdün kuyruğuna geldin’, which can have two meanings because of the word ‘yüzmek’. Namely, it may mean that one is skinning a slaughtered sheep and has reached its tail, while the other meaning of this sentence would be that one has begun an action, ‘yüzmek’ (swim), and is about to finish

the action. The contribution of this dissertation to the literature is its novel hybrid algorithm. However, there are still some improvements to be made to the proposed approaches:

- To disambiguate all ambiguous words using language modeling based on neural networks. We want to use ANN-based language modeling for this challenging task because these systems are able to preserve the contextual neighbors at long distances, while running forward and backward. Since the ANN-based LSTM model learns to correct the errors in the ground truth by incorporating context-aware processing [4], the real-word error problem can be solved.

- To extend the XML construction in Chapter 1 in Figure 1.1 in order to build a structured data. To develop some algorithms to identify topics from classified words using some word sense disambiguation techniques to study domain classification using these constructed large corpora. We will classify words with same meaning by labeling words.

- To build a large structured XML corpora. Large corpora serve as important research tools for investigators in natural language processing and speech recognition, as well as theoretical linguistics [110].

- To correct possible real-word and non-word errors.

- To apply the model to a large big data set from OCRd text.

- To apply our new model to other languages.

REFERENCES

[1] Adnan Ul-Hasan. “Generic text recognition using long short-term memory networks”. In: (2016).

[2] Mai Oudah & Khaled Shaalan. “A pipeline Arabic named entity recognition using a hybrid approach”. In: *Proceedings of COLING 2012*. 2012, pp. 2159–2176.

[3] Sepp Hochreiter & Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[4] Adnan Ul-Hasan, Syed Saqib Bukhari & Andreas Dengel. “Ocroract: A sequence learning ocr system trained on isolated characters”. In: *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. IEEE. 2016, pp. 174–179.

[5] Karen Kukich. “Techniques for automatically correcting words in text”. In: *Acm Computing Surveys (CSUR)* 24.4 (1992), pp. 377–439.

[6] U Pal, Pulak K Kundu & Bidyut Baran Chaudhuri. “OCR error correction of an inflectional indian language using morphological parsing”. In: *J. Inf. Sci. Eng.* 16.6 (2000), pp. 903–922.

[7] Tomáš Mikolov et al. “Statistical language models based on neural networks”. In: *Presentation at Google, Mountain View, 2nd April* 80 (2012), p. 26.

[8] U Pal & BB Chaudhuri. “Indian script character recognition: a survey”. In: *pattern Recognition* 37.9 (2004), pp. 1887–1899.

[9] Michele Banko & Eric Brill. “Scaling to very very large corpora for natural language disambiguation”. In: *Proceedings of the 39th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2001, pp. 26–33.

[10] Youssef Bassil & Mohammad Alwani. “Ocr post-processing error correction algorithm using google online spelling suggestion”. In: *arXiv preprint arXiv:1204.0191* (2012).

[11] Kanwar Abrar Ahmad, Muhammad Munwar Iqbal & Yousaf Saeed. “Error Detection and Correction in Nlp Using Finite State Automata: Urdu Text Processing”. In: (2014).

[12] ABBYY FineReader Engine. “URL: https://www. abbyy. com/en-us/ocr-sdk”. In: *Accessed on March* 14 (2019).

[13] Ray Smith. “An overview of the Tesseract OCR engine”. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 2. IEEE. 2007, pp. 629–633.

[14] Kurtuluğ Karasu & Muhammet Bağtan. “Turkish OCR on mobile and scanned document images”. In: *2015 23nd Signal Processing and Communications Applications Conference (SIU)*. IEEE. 2015, pp. 2074–2077.

[15] Burcu Can. “Unsupervised learning of allomorphs in Turkish”. In: *Turkish Journal of Electrical Engineering & Computer Sciences* 25.4 (2017), pp. 3253–3260.

[16] Evan L Antworth. “PC-KIMMO: a two-level processor for morphological analysis”. In: *Summer Institute of Linguistics* (1990).

[17] Kenneth R Beesley & Lauri Karttunen. “Finite-state morphology: Xerox tools and techniques”. In: *CSLI, Stanford* (2003).

[18] Helmut Schmid. “A programming language for finite state transducers”. In: *FSMNLP*. Vol. 4002. 2005, pp. 308–309.

[19] Kemal Oflazer. “Two-level description of Turkish morphology”. In: *Literary and linguistic computing* 9.2 (1994), pp. 137–148.

[20] Jorge Hankamer. “Finite state morphology and left-to-right morphology”. In: *West Coast Conference on Formal Linguistics, Bildiri Kitab*. 1986.

[21] Helmut Schmid, Arne Fitschen & Ulrich Heid. “SMOR: A German Computational Morphology Covering Derivation, Composition and Inflection.” In: *LREC*. Lisbon. 2004, pp. 1–263.

[22] BB Chaudhuri & U Pal. “A complete printed Bangla OCR system”. In: *Pattern recognition* 31.5 (1998), pp. 531–549.

[23] Eva D’hondt, Cyril Grouin & Brigitte Grau. “Generating a training corpus for OCR post-correction using encoder-decoder model”. In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2017, pp. 1006–1014.

[24] Rohit Saluja, Devaraj Adiga, Parag Chaudhuri, Ganesh Ramakrishnan & Mark Carman. “Error detection and corrections in Indic OCR using LSTMs”. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 1. IEEE. 2017, pp. 17–22.

[25] Yulia Tsvetkov. “Linguistic Knowledge in Data-Driven Natural Language Processing”. PhD thesis. PhD thesis, Carnegie Mellon University, 2016.

[26] Nal Kalchbrenner & Phil Blunsom. “Recurrent continuous translation models”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013, pp. 1700–1709.

[27] Helmut Schmid. “Part-of-speech tagging with neural networks”. In: *arXiv preprint cmp-lg/9410018* (1994).

[28] Alex Graves, Navdeep Jaitly & Abdel-rahman Mohamed. “Hybrid speech recognition with deep bidirectional LSTM”. In: *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE. 2013, pp. 273–278.

[29] Minh-Thang Luong & Christopher D Manning. “Achieving open vocabulary neural machine translation with hybrid word-character models”. In: *arXiv preprint arXiv:1604.00788* (2016).

[30] Kemal Oflazer. “Error-tolerant finite state recognition with applications to morphological analysis and spelling correction”. In: *arXiv preprint cmp-lg/9504031* (1995).

[31] Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong & Jianfeng Gao. “A nested attention neural hybrid model for grammatical error correction”. In: *arXiv preprint arXiv:1707.02026* (2017).

[32] Mucahid Kutlu & Ilyas Cicekli. “A hybrid morphological disambiguation system for turkish”. In: *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. 2013, pp. 1230–1236.

[33] Gülşen Eryiğit & Kemal Oflazer. “Statistical dependency parsing for turkish”. In: *11th Conference of the European Chapter of the Association for Computational Linguistics*. 2006.

[34] Deniz Yuret & Ferhan Türe. “Learning morphological disambiguation rules for Turkish”. In: *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics. 2006, pp. 328–334.

[35] Turhan Daybelge & Ilyas Cicekli. “A rule-based morphological disambiguator for Turkish”. In: *Proceedings of Recent Advances in Natural Language Processing*. 2007, pp. 145–149.

[36] Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell & Chris Dyer. “The role of context in neural morphological disambiguation”. In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016, pp. 181–191.

[37] Eray Yildiz, Caglar Tirkaz, H Bahadır Sahin, Mustafa Tolga Eren & Omer Ozan Sonmez. “A morphology-aware network for morphological disambiguation”. In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[38] Jakub Náplava. “Natural language correction”. In: (2017).

[39] François Chollet et al. “Keras: The python deep learning library”. In: *ascl* (2018), ascl–1806.

[40] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, et al. “Tensorflow: A system for large-scale machine learning”. In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 2016, pp. 265–283.

[41] Adnan Ul-Hasan & Thomas M Breuel. “Can we build language-independent OCR using LSTM networks?” In: *Proceedings of the 4th International Workshop on Multilingual OCR*. 2013, pp. 1–5.

[42] Ahmet Afsin Akın & Mehmet Dündar Akın. “Zemberek, an open source nlp framework for turkic languages”. In: *Structure* 10 (2007), pp. 1–5.

[43] Alan Woodley, Xavier Tannier, Marcus Hassler & Shlomo Geva. “Natural Language Processing and XML Retrieval”. In: *Proceedings of the 2006 Australasian Language Technology Workshop*. University of Sydney. 2006, pp. 165–166.

[44] Christopher D Manning, Christopher D Manning & Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[45] Dirk Goldhahn, Thomas Eckart & Uwe Quasthoff. “Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages.” In: *LREC*. Vol. 29. 2012, pp. 31–43.

[46] Eduard Hovy & Julia Lavid. “Towards a ‘science’of corpus annotation: a new methodological challenge for corpus linguistics”. In: *International journal of translation* 22.1 (2010), pp. 13–36.

[47] Haşim Sak, Tunga Güngör & Murat Saraçlar. “Morphological disambiguation of Turkish text with perceptron algorithm”. In: *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer. 2007, pp. 107–118.

[48] Karen Sparck Jones. “Natural language processing: a historical review”. In: *Current issues in computational linguistics: in honour of Don Walker*. Springer, 1994, pp. 3–16.

[49] Steven Bird, Ewan Klein & Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O’Reilly Media, Inc.", 2009.

[50] M Oguzhan Külekcỳ & Mehmed Özkan. “Turkish word segmentation using morphological analyzer”. In: *Seventh European Conference on Speech Communication and Technology*. 2001.

[51] Kemal Oflazer & Sharon Inkelas. “A finite state pronunciation lexicon for Turkish”. In: *Proceedings of the EACL Workshop on Finite State Methods in NLP*. Vol. 82. 2003, pp. 900–918.

[52] GÜLSEN ERYI, DILARA TORUNO, et al. “Social media text normalization for Turkish”. In: *Natural Language Engineering* 23.6 (2017), p. 835.

[53] Tuğba Pamay, Umut Sulubacak, Dilara Torunoğlu-Selamet & Gülşen Eryiğit. “The annotation process of the ITU Web Treebank”. In: *Proceedings of the 9th Linguistic Annotation Workshop*. 2015, pp. 95–101.

[54] Andrej Karpathy. “The unreasonable effectiveness of recurrent neural networks”. In: *Andrej Karpathy blog* 21 (2015), p. 23.

[55] AYLA KAYABAŞ, Helmut Schmid, AHMET ERCAN TOPCU & ÖZKAN KILIÇ. “TRMOR: a finite-state-based morphological analyzer for Turkish”. In: *Turkish Journal of Electrical Engineering & Computer Sciences* 27.5 (2019), pp. 3837–3851.

[56] Lauri Karttunen. “Finite-state constraints”. In: *The last phonological rule* 6 (1993), pp. 173–194.

[57] Fabienne Cap. “Morphological processing of compounds for statistical machine translation”. In: (2014).

[58] Philipp Koehn. “Europarl: A parallel corpus for statistical machine translation”. In: *MT summit*. Vol. 5. Citeseer. 2005, pp. 79–86.

[59] Aslı Göksel & Celia Kerslake. *Turkish: A comprehensive grammar*. Routledge, 2004.

[60] Jorge Hankamer. “Turkish vowel epenthesis”. In: *Puzzles of languages: Essays in honour of Karl Zimmer* (2011), pp. 55–69.

[61] Mans Hulden. “Foma: a finite-state compiler and library”. In: *Proceedings of the Demonstrations Session at EACL 2009*. 2009, pp. 29–32.

[62] Krister Lindén, Miikka Silfverberg & Tommi Pirinen. “Hfst tools for morphology–an efficient open-source package for construction of morphological analyzers”. In: *International Workshop on Systems and Frameworks for Computational Morphology*. Springer. 2009, pp. 28–47.

[63] Cagri Cöltekin. “A Freely Available Morphological Analyzer for Turkish.” In: *LREC*. Vol. 2. 2010, pp. 19–28.

[64] Uwe Springmann, Helmut Schmid & Dietmar Najock. “LatMor: A Latin finite-state morphology encoding vowel quantity”. In: *Open Linguistics* 2.1 (2016).

[65] Daniel Karp, Yves Schabes, Martin Zaidel & Dania Egedi. “A freely available wide coverage morphological analyzer for English”. In: *Proceedings of the 14th conference on Computational linguistics-Volume 3*. Association for Computational Linguistics. 1992, pp. 950–955.

[66] Lauri Karttunen & Kenneth R Beesley. “A short history of two-level morphology”. In: *ESSLLI-2001 Special Event titled” Twenty Years of Finite-State Morphology* (2001).

[67] Aysin Solak & Kemal Oflazer. “Parsing agglutinative word structures and its application to spelling checking for Turkish”. In: *Proceedings of the 14th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics. 1992, pp. 39–45.

[68] Arne Fitschen. “Ein computerlinguistisches Lexikon als komplexes System”. PhD thesis. Inst. für Maschinelle Sprachverarbeitung der Univ., 2004.

[69] Harry Van der Hulst & Jeroen Van De Weijer. “Topics in Turkish phonology”. In: *Turkish linguistics today* (1991), pp. 11–59.

[70] S Murat Oztaner. “A word grammar of Turkish with morphophonemic rules”. In: *arXiv preprint cmp-lg/9608013* (1996).

[71] Kai-Uwe Carstensen, Christian Ebert, Cornelia Ebert, Susanne Jekat, Hagen Langer & Ralf Klabunde. *Computerlinguistik und Sprachtechnologie: Eine Einführung*. Springer-Verlag, 2009.

[72] Mikko Kurimo, Mathias Creutz & Matti Varjokallio. “Unsupervised Morpheme Analysis Evaluation by a Comparison to a Linguistic Gold Standard-Morpho Challenge 2007.” In: *CLEF (Working Notes)*. 2007.

[73] Gertrud Faaß, Ulrich Heid & Helmut Schmid. “Design and Application of a Gold Standard for Morphological Analysis: SMOR as an Example of Morphological Evaluation.” In: *LREC*. 2010.

[74] Mayce Al Azawi. “Statistical Language Modeling for Historical Documents using Weighted Finite-State Transducers and Long Short-Term Memory”. In: (2015).

[75] Chunting Zhou, Chonglin Sun, Zhiyuan Liu & Francis Lau. “A C-LSTM neural network for text classification”. In: *arXiv preprint arXiv:1511.08630* (2015).

[76] Alex Graves & Navdeep Jaitly. “Towards end-to-end speech recognition with recurrent neural networks”. In: *International conference on machine learning*. 2014, pp. 1764–1772.

[77] Haşim Sak, Andrew Senior & Françoise Beaufays. “Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition”. In: *arXiv preprint arXiv:1402.1128* (2014).

[78] Gülin Dede & Murat Hüsnü Sazlı. “Speech recognition with artificial neural networks”. In: *Digital Signal Processing* 20.3 (2010), pp. 763–768.

[79] Kh Tohidul Islam, Ghulam Mujtaba, Ram Gopal Raj & Henry Friday Nweke. “Handwritten digits recognition with artificial neural network”. In: *2017 International Conference on Engineering Technology and Technopreneurship (ICE2T)*. IEEE. 2017, pp. 1–4.

[80] Vivek Shrivastava & Navdeep Sharma. “Artificial neural network based optical character recognition”. In: *arXiv preprint arXiv:1211.4385* (2012).

[81] Guillaume Chiron, Antoine Doucet, Mickaël Coustaty & Jean-Philippe Moreux. “Icdar2017 competition on post-ocr text correction”. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 1. IEEE. 2017, pp. 1423–1428.

[82] Taeho Jo. “NTC (Neural Text Categorizer): neural network for text categorization”. In: *International Journal of Information Studies* 2.2 (2010), pp. 83–96.

[83] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami & Chris Dyer. “Neural architectures for named entity recognition”. In: *arXiv preprint arXiv:1603.01360* (2016).

[84] Li Dong & Mirella Lapata. “Coarse-to-fine decoding for neural semantic parsing”. In: *arXiv preprint arXiv:1805.04793* (2018).

[85] Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li & Xiaoming Li. “Neural generative question answering”. In: *arXiv preprint arXiv:1512.01337* (2015).

[86] Jian Tang, Yifan Yang, Sam Carton, Ming Zhang & Qiaozhu Mei. “Context-aware natural language generation with recurrent neural networks”. In: *arXiv preprint arXiv:1611.09900* (2016).

[87] Ziqiang Cao, Furu Wei, Li Dong, Sujian Li & Ming Zhou. “Ranking with Recursive Neural Networks and Its Application to Multi-Document Summarization.” In: *AAAI*. Citeseer. 2015, pp. 2153–2159.

[88] S Sooraj, K Manjusha, M Anand Kumar & KP Soman. “Deep learning based spell checker for Malayalam language”. In: *Journal of Intelligent & Fuzzy Systems* 34.3 (2018), pp. 1427–1434.

[89] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende & Daan Wierstra. “Draw: A recurrent neural network for image generation”. In: *arXiv preprint arXiv:1502.04623* (2015).

[90] Aminul Islam & Diana Inkpen. “Real-word spelling correction using google web 1tn-gram data set”. In: *Proceedings of the 18th ACM conference on Information and knowledge management*. 2009, pp. 1689–1692.

[91] VS Vinitha & CV Jawahar. “Error detection in indic ocrs”. In: *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. IEEE. 2016, pp. 180–185.

[92] Kareem Mokhtar, Syed Saqib Bukhari & Andreas Dengel. “OCR Error Correction: State-of-the-Art vs an NMT-based Approach”. In: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE. 2018, pp. 429–434.

[93] Alex Graves. “Long short-term memory”. In: *Supervised sequence labelling with recurrent neural networks*. Springer, 2012, pp. 37–45.

[94] Andrej Karpathy. “Connecting images and natural language”. PhD thesis. Ph. D. thesis, Stanford University, 2016.

[95] Zhuohan Li, Di He, Fei Tian, Wei Chen, Tao Qin, Liwei Wang, et al. “Towards binary-valued gates for robust lstm training”. In: *arXiv preprint arXiv:1806.02988* (2018).

[96] Alex Graves. “Generating sequences with recurrent neural networks”. In: *arXiv preprint arXiv:1308.0850* (2013).

[97] Alex Graves, Santiago Fernández & Jürgen Schmidhuber. “Bidirectional LSTM networks for improved phoneme classification and recognition”. In: *International Conference on Artificial Neural Networks*. Springer. 2005, pp. 799–804.

[98] Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Y Ng & Dan Jurafsky. “Noising and denoising natural language: Diverse backtranslation for grammar correction”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018, pp. 619–628.

[99] Andrej Karpathy, Justin Johnson & Li Fei-Fei. “Visualizing and understanding recurrent networks”. In: *arXiv preprint arXiv:1506.02078* (2015).

[100] Ilya Sutskever, Oriol Vinyals & Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[101] Yoav Goldberg. “Neural network methods for natural language processing”. In: *Synthesis Lectures on Human Language Technologies* 10.1 (2017), pp. 1–309.

[102] S. Göker & B. Can. “Neural Text Normalization for Turkish Social Media”. In: *2018 3rd International Conference on Computer Science and Engineering (UBMK)*. 2018, pp. 161–166.

[103] O Kilic. “ÖZKAN KILIÇ”. In: *Instructor* 2008 (2004).

[104] Ehud Reiter & Robert Dale. *Building natural language generation systems*. Cambridge university press, 2000.

[105] Jie Mei, Aminul Islam, Yajing Wu, Abidalrahman Moh’d & Evangelos E Milios. “Statistical learning for OCR text correction”. In: *arXiv preprint arXiv:1611.06950* (2016).

[106] Richard Sproat & Navdeep Jaitly. “RNN approaches to text normalization: A challenge”. In: *arXiv preprint arXiv:1611.00068* (2016).

[107] Ekin Akyürek, Erenay Dayanık & Deniz Yuret. “Morphological analysis using a sequence decoder”. In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 567–579.

[108] Miikka Silfverberg & Jack Rueter. “Can Morphological Analyzers Improve the Quality of Optical Character Recognition?” In: *Septentrio Conference Series*. 2. 2015, pp. 45–56.

[109] Okan Kolak & Philip Resnik. “OCR error correction using a noisy channel model”. In: *Proceedings of the second international conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc. 2002, pp. 257–262.

[110] Mitchell Marcus, Beatrice Santorini & Mary Ann Marcinkiewicz. “Building a large annotated corpus of English: The Penn Treebank”. In: (1993).

CURRICULUM VITAE

PERSONAL INFORMATION

Name Surname: Ayla Kayabaş

EDUCATION

Degree	Institution	Year of Graduation
B.Sc.	University of Stuttgart	2011
M.Sc.	University of Stuttgart	2011

PUBLICATION(S)

Published

KAYABAŞ, A., Schmid, H., TOPCU, A. E., & KILIÇ, Ö. 2019. TRMOR: a finite-state-based morphological analyzer for Turkish. Turkish Journal of Electrical Engineering & Computer Sciences, 27(5): 3837–3851.

In Review

KAYABAŞ, A., TOPCU, A. E., KILIÇ, Ö. 2020. A Novel Hybrid Algorithm for Language Modeling: Artificial Neural-Net-XMOR. The Journal of Natural Language Engineering, Cambridge University Press.