JOINT LEARNING OF MORPHOLOGICAL SEGMENTATION, MORPHEME TAGGING, PART-OF-SPEECH TAGGING, AND DEPENDENCY PARSING

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

HÜSEYIN ALEÇAKIR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COGNITIVE SCIENCE

JANUARY 2020

Approval of the thesis:

JOINT LEARNING OF MORPHOLOGICAL SEGMENTATION, MORPHEME TAGGING, PART-OF-SPEECH TAGGING, AND DEPENDENCY PARSING

submitted by **HÜSEYIN ALEÇAKIR** in partial fulfillment of the requirements for the degree of **Master of Science in Cognitive Science, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek
Dean, **Graduate School of Informatics**

Prof. Dr. Cem Bozşahin
Head of Department, **Cognitive Science, METU**

Prof. Dr. Cem Bozşahin
Supervisor, **Cognitive Science, METU**

Assist. Prof. Dr. Burcu Can Buğlalılar
Co-supervisor, **Department of Computer Engineering, Hacettepe University**

Examining Committee Members:

Prof. Dr. Deniz Zeyrek
Cognitive Science Department, METU

Prof. Dr. Cem Bozşahin
Cognitive Science Department, METU

Assist. Prof. Dr. Burcu Can Buğlalılar
Department of Computer Engineering, Hacettepe University

Prof. Dr. İlyas Çiçekli
Department of Computer Engineering, Hacettepe University

Assist. Prof. Dr. Umut Özge
Cognitive Science Department, METU

Date:

ABSTRACT

JOINT LEARNING OF MORPHOLOGICAL SEGMENTATION, MORPHEME TAGGING, PART-OF-SPEECH TAGGING, AND DEPENDENCY PARSING

Aleçakır, Hüseyin
M.S., Department of Cognitive Science
Supervisor : Prof. Dr. Cem Bozşahin
Co-Supervisor : Assist. Prof. Dr. Burcu Can Buğlalılar

January 2020, 45 pages

In agglutinating languages, there is a strong relationship between morphology and syntax. Inflectional and derivational suffixes have a significant role while determining the syntactic role of the word in the sentence. This connection enables the joint learning of morphology and syntax. Apart from that, the complex morphology poses a sparsity problem. In this respect, morphological analysis and segmentation are vital for various natural language processing applications. All of these have provided the primary motivation to develop a joint learning model for morphological segmentation, morphological tagging, part-of-speech (POS) tagging, and dependency parsing. The proposed model consists of a multi-layered neural network structure. In each level, there is a bidirectional long-short memory unit (BiLSTM) to encode sequential information. Additionally, attention networks are used to compute soft alignment between encoder-decoder states in the morphological tagging component. Finally, the obtained results from each layer of the network are compared with other works from the literature. The results are very competitive on Universal Dependencies (UD) dataset.

Keywords: Dependency Parsing, Part of Speech Tagging, Syntax, Morphology, Deep Learning

ÖZ

MORFOLOJİK ANALİZ, SÖZCÜK TÜRÜ İŞARETLEME VE BAĞLILIK AYRIŞTIRMANIN EŞ ZAMANLI ÖĞRENİLMESİ

Aleçakır, Hüseyin
Yüksek Lisans, Bilişsel Bilimler Programı
Tez Yöneticisi : Prof. Dr. Cem Bozşahin
Ortak Tez Yöneticisi : Yrd. Doç. Dr. Burcu Can Buğlalılar

Ocak 2020 , 45 sayfa

Bitişimli dillerde biçimbilim ve sözdizim arasında güçlü bir ilişki vardır. Çekim ve yapım eklerini sözcüğün sözdizimsel rolünün belirlenmesinde etkili olmaktadır. Sözdizim ve biçimbilim arasındaki bu bağlantı ikisinin eşzamanlı öğrenilmesini olanaklı kılmaktadır. Bunun yanında dildeki karmaşık morfolojinin varlığı seyreklik problemine neden olabilmektedir. Bu nedenden ötürü morfolojik analiz ve işaretmele pek çok doğal dil işleme çalışması için zaruridir. Yukarıda sıralanan sebepler morfolojik analiz ve işaretleme, sözcük türlerinin bulunması ve bağlılık ayrıştırma gibi farklı sözdizimsel analizlerin yapılmasına teşvik etmektedir. Önerilen model çok katmanlı yapay sinir ağlarından oluşmaktadır. Her bir seviyede ardışık bilgiyi kodlayabilen iki yönlü uzun kısa vadeli bellek ağı birimleri bulunmaktadır. Buna ilaveten, morfolojik etiketleme bileşeninde kullanılan dikkat ağı yardımıyla kodlayıcı ve çözücü durumları arasındaki hizanlamanın kurulması amaçlandı. Her bir seviyeden elde edilen sonuçlar literatürdeki diğer çalışmalarla karşılaştırıldı. Evrensel Bağlılık Ağaç Yapılı veri kümesinde karşılaştırabilinir sonuçlar alındı.

Anahtar Kelimeler: Bağlılık Ayrıştırma, Sözcük Türü İşaretleme, Sözdizimi, Morfoloji, Derin Öğrenme

ACKNOWLEDGMENTS

I thank my supervisor Cem Bozsahin for his guidance and support. Furthermore, I thank my co-supervisor Burcu Can Buglalilar for introducing me to the topic as well for the support on the way.

I thank my friends for being there whenever I need them.

I thank Naz for every single moment we had together.

Finally, I thank my family for always being there for me, whatever it takes.

This study is partially funded by TUBITAK (The Scientific and Technological Research Council of Turkey) with grant number 115E464.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BI-LSTM	Bidirectional Long short-term memory
BTS	Byte-to-Span
CRF	Conditional Random Field
HMM	Hidden Markov Model
LSTM	Long Short-Term Memory
MAP	Maximum a Posteriori
MDL	Minimum Description Length
MEMM	Maximum Entropy Markov Model
MLP	Multi Layer Perceptron
NLP	Natural Language Processing
RNN	Recurrent Neural Network
POS	Part Of Speech
UD	Universal Dependencies

CHAPTER 1

INTRODUCTION

The subfields of linguistics concern with different levels and aspects of the structure of language. There are multiple levels of structure of language to deal with interaction with cognition. Furthermore, the different components of language work together to convey the intended meaning. From the natural language processing (NLP) perspective, to able to achieve human-level language understanding, various aspects of language must be taken into consideration.

Most of the NLP systems pay attention to tasks related to text-based input. Almost all of these tasks require word-level and sentence-level syntactic information to extract the representation of who did what to whom. In some cases, word-level information can constrain the combinatoric potential of words at the sentence level. It is also frequent in languages with rich morphology. NLP systems have to incorporate word-level information to achieve linguistically plausible syntactic models.

The focus of this study will be on morphology and syntax. Morphology is the subfield of linguistics. In broad terms, it can be defined as the study of word formation and concerned with the internal structure of words. The unit under consideration is a morpheme, which is the smallest meaning-bearing units of language. The syntax of a language can be viewed as a system of rules for constructing grammatical sentences out of words and determining their form and meaning. Morphology and syntax work together to convey the intended meaning. This interaction enables the joint learning of these syntactic units.

Morphological segmentation and morphological tagging are word-level syntactic tasks that deal with morphemes in word, and part-of-speech (POS) tagging and dependency parsing are sentence-level syntactic tasks.

The goal of morphological segmentation is to break a word into morphemes in a particular context. Some of the work in the literature deals with morphology as a segmentation task that only aims to segment each word into its morphemes (Harris, 1955; Yarowsky & Wicentowski, 2000; Goldsmith, 2001; Creutz & Lagus, 2002, 2005a; Can & Manandhar, 2013; Soricut & Och, 2015; Üstün & Can, 2016). In morphological tagging, given a surface form in context, only sequences of morphological features predicted. Some of the work deals with morphology as both segmentation and sequence labeling problem, which is called morphological tagging (Müller et al., 2013; Heigold et al., 2016; Dayanik, 2018). POS tagging investigates the syntactic functions of words in the sentence. Thus far, several versions of POS tagger are proposed by Church (1989); DeRose (1988); Schütze & Singer (1994); Brants (2000); Tseng et al. (2005); Gillick et al. (2015); Ling et al. (2015); Plank et al. (2016). Dependency parsing describes the internal structure of the sentence in terms of directed relations between words in the sentence. In the last few decades, there has been a surge of interest in supervised methods for data-driven dependency parsing (Kudo & Matsumoto, 2002; Yamada & Matsumoto, 2003; Eisner, 1996; McDonald et al., 2005).

Recent work shows that representation that is learned by using words as separate tokens are not well enough to capture the syntactic and semantic information of words in agglutinative languages because of two main reasons. One reason is that various kinds of information are expressed in morphology. For instance, tense, aspect, and mood are commonly expressed via inflectional morphology. These morphological features are the grammaticalized properties of events. The second reason is that morphological complexity increases data sparsity. Hence, words as separate tokens are not well enough to capture the syntactic and semantic information of words in agglutinative languages.

In this study, a joint model for learning morphological segmentation, morpheme tagging, POS tagging, and dependency parsing, especially in agglutinating languages, is introduced. The four tasks can be learned efficiently in a single unified framework by helping each other through sharing the neural vector representations between layers. As a consequence of this hierarchal sharing mechanism, upper layers, such as dependency parsing or POS tagging, can access morphological information that are encoded in words. Here, morpheme and character representations have been used to cope with the sparsity problem.

Our framework is built upon the joint PoS tagging and dependency parsing model of Nguyen & Verspoor (2018), where we introduce two more layers for morphological segmentation and morphological tagging. Therefore, this study can be seen as an extension of Nguyen & Verspoor (2018) and Kiperwasser & Goldberg (2016).

To our knowledge, there has not been any study that combines morphological segmentation, morpheme tagging, POS tagging, and dependency parsing in a single framework using deep neural networks, especially for morphologically rich languages.

The results show that such a unified framework benefits from joint learning of various levels from morphology to dependencies significantly.

The thesis is divided into five distinct sections. The remaining parts of this study are structured as follows. Chapter 2 establishes the linguistics background of morphology and syntax and summarizes the related work on morphological segmentation, morpheme tagging, part-of-speech (POS) tagging, dependency parsing, and joint learning. Chapter 3 gives the mathematical background of sequence-based neural networks and details the proposed model. Chapter 4 explains data, implementation details, and analyzes the results obtained from experiments. Chapter 5 concludes the study with future goals.

CHAPTER 2

RELATED WORK

2.1 Morphological Segmentation

Morphology is the subfield of linguistics. In broad terms, it can be defined as the study of word formation and concerned with the internal structure of words. The unit under consideration is a morpheme, which is the smallest meaning-bearing units of language. The goal of morphological segmentation is to break a word into morphemes in a particular context.

The first heuristics-based unsupervised morpheme segmentation method (LSV) was first presented by (Harris, 1955) and further investigated by others. Harris (1970) proposed a segmentation algorithm, which is based on distributional characteristics of letters in utterance, to detect possible morpheme boundaries of a word. In the 2000s, there are many studies that follow the unsupervised tradition for morphology learning (Yarowsky & Wicentowski, 2000; Goldsmith, 2001; Creutz & Lagus, 2002, 2005a; Can & Manandhar, 2013).

Goldsmith's Linguistica (Goldsmith, 2001) was based on information-theoretic principle, which is called Minimum Description Length (MDL). There is some equivalence between the notion of probability in Bayesian learning and the notion of length of a binary message in information theory. One needs to maximize the posterior probability in Bayesian learning and minimize the message length in information theory. There is an important notion called a signature introduced by Linguistica. Signatures are a set of affixes attached to given stems; these signatures are collected from an unannotated corpus. Thus, Linguistica represents the language of morphology by signatures and signature-stem pairs. To choose the best hypothesis to explain the data, Linguistica makes use of MDL for stem-signature coding.

Yarowsky & Wicentowski (2000) introduced a minimally supervised corpus-based morphological analyzer. The proposed algorithm learns the alignment between roots and their inflected forms. Unlike Goldsmith (2001), this method can learn non-affixal alignments forms such as $< take, took >$.

Creutz & Lagus (2002) proposed two methods for learning of morphological segmentation: MDL-based and probabilistic-based. There are two parts of the data representation in this model. The first one is called codebook—dictionary of morphemes; the second one is a sequence of texts. The cost function of the MDL-based model is as follows:

$$\begin{aligned} Totalcost &= cost(source) + cost(codebook) \\ &= \sum_{m_i \in \mathbb{T}} -\log p(m_i) + \sum_{m_j \in \mathbb{M}} k * length() \end{aligned} \tag{2.1}$$

where $\mathbb{M}$ and $\mathbb{T}$ denote the number of morpheme types and sequence of tokens, respectively. The function math $length(m_j)$ computes the length of token m_j in characters. k is the maximum code length of a character. The negative log-likelihood is used to compute the cost of a sequence of tokens. This cost function aims to reduce the size of the codebook.

As opposed to the MDL-based model, the second model of (Creutz & Lagus, 2002) uses the maximum likelihood of data without prior information. Here, codebook cost is not included:

$$\begin{aligned} Totalcost &= cost(source) \\ &= \sum_{m_i \in \mathbb{T}} -\log p(m_i) \end{aligned} \tag{2.2}$$

where the total cost is equal to the summation of all tokens' negative log probabilities. A variant of expectation-maximization is used to estimate the model parameters.

Creutz & Lagus (2005a,b) presented another Morfessor family tool based on (Creutz & Lagus, 2002) for unsupervised segmentation. The Morfessor relies on the priors about morphemes such as length, frequency, and perplexity. The maximum a posteriori (MAP) estimation is employed during the training step to determine optimal lexicon and segmentation.

$$\underset{lexion}{\operatorname{argmax}} P(lexicon|corpus) = \underset{lexicon}{\operatorname{argmax}} P(corpus|lexicon) * P(lexicon) \tag{2.3}$$

where *lexicon* and *corpus* correspond to a set of distinct morphemes and sequences of tokens, respectively.

Can & Manandhar (2013) presented a Dirichlet process model for joint POS tagging and morphological segmentation. It is a generative model where the authors generate the POS tags, stems, and suffixes of words jointly. A mixture model is adopted for POS tagging by using the tags of the contextual words. A Dirichlet process model is adopted for morphology learning where stems may belong to any POS tag. The model generates stems based on POS tags. It is a mixture model. Moreover, It adopts a Dirichlet process model for morphology-based on POS tags.

A great deal of previous research on morphological segmentation has focused on the relationship between morphology and semantics (Soricut & Och, 2015; Üstün & Can, 2016). These studies have assessed the efficacy of distribution-based word vector representations such as the Skipgram model Mikolov et al. (2013) and subword model Bojanowski et al. (2017). Skipgram model Mikolov et al. (2013) uses neighbor words to estimate word vectors. It encodes words to dense vectors using word co-occurrence statistics.

Üstün & Can (2016) proposed a segmentation algorithm based on the skipgram model. The model proposed by (Üstün & Can, 2016) takes the input pair, which has the form of (word:

logical form) and using the semantic vector knowledge, it reduces the number of possible combinations to segmentation set called pseudo-morphemes. After producing all possible combinations, the model makes use of semantic vector information to eliminate segmentations below a given threshold. The model proposed by (Üstün & Can, 2016) calculates the cosine distance between word and its suffixed form, e.g., cosine_similarity(araba, arabalar), and repeats this process for each split point in the word. The primary assumption of Üstün & Can (2016) is that inflectional morphemes do not change the meaning of word radically, so cosine similarity between word itself and its inflected form has to be higher than 0.25. However, this proposed method is not directly applicable to the derivational morphology.

Soricut & Och (2015) introduced the language-agnostic unsupervised Transformation-Based segmentation model. Soricut & Och (2015) shows that morphological representations can be extracted using semantic vectors. This work only considers prefixes and suffixes. The below pipeline overviews the proposed algorithm.

Given a finite vocabulary V:

1. Extract candidate prefix/suffix rules from V
2. Train embedding space $E^n \subset \mathbb{R}$ for all words in V
3. Evaluate the quality of candidate rules in E^n

4.Generate lexicalized morphological transformations

In the first phase, it finds all tuples, i.e. *candidate rules*, $(w_1, w_2) \in V$ starting with the same prefix or ends with the same substring. Then, in the second stage, they train all words in V using Skipgram model.

type:from:to	$(\mathbf{w_1}, \mathbf{w_2})$
suffix:ed:ing	$\Rightarrow$ (increased, increasing)
prefix:un:re	$\Rightarrow$ (unmade, remade)
suffix:ion:ed	$\Rightarrow$ (creation, created)

In the following phases, Soricut & Och (2015) evaluated the quality of candidate rules using the *meaning-preservation property*, and quantitatively measured the assertion *'car is to cars what dog is to dogs'*. Furthermore, the last step, to avoid over-applying, e.g., suffix:ε:ly $\Rightarrow$ (only,ly), they restrict the application by graph reduction. This model outperformed past works on word similarity tasks.

2.2 Morphological Tagging

A considerable amount of information is encoded within words in morphologically rich languages. Morphologic analyzers and disambiguators identify the underlying form of words– morphemes and morphological features. Some of the NLP applications require this information encoded in a word, e.g., machine translation models. Traditionally, morphologic analyzers typically are carried out by language-specific fine-state transducers. They can produce possible analysis, and then morphological disambiguators assign particular analysis to a surface string within context. TRMorph for Turkish (Cöltekin, 2010) and Morphisto for German (Zielinski & Simon, 2009) are open-source implementations of two-level morphological analyzers.

The task of morphological tagging is similar to morphological analysis. In morphological tagging, given a surface form in context, only sequences of morphological features predicted. Also, several studies (Müller et al., 2013; Heigold et al., 2016; Dayanik, 2018) incorporate part-of-speech tag into sequences of predicted features. It is important to note that the morphological tagger proposed in this study excludes part-of-speech tags.

Müller et al. (2013) introduced a Conditional Random Fields (CRFs) (Lafferty et al., 2001) based sequence prediction model for morphological tagging. Thus far, several studies have begun to examine the use of neural networks in the task of morpheme tagging (Heigold et al., 2016; Dayanik, 2018; Malaviya et al., 2018). The first of them is proposed by Heigold et al. (2016). In this study, two character-based morpheme segmentation models was compared—convolutional neural networks (CNN) and long short term memory networks. Heigold et al. (2016) have shown that LSTM architectures are more suitable for morphological tasks. Cotterell & Heigold (2017) subsequently improved a character-based LSTM model with the transfer learning techniques. They aimed to transfer the obtained knowledge from high-resource language to low-resource languages.

Finally, MorphNet (Dayanik, 2018) made use of sequence-to-sequence neural architectures to combine morphological analysis and disambiguation in a single model. MorphNet has three recurrent neural networks (RNNs) to encode features of input words on different levels: word encoder, context encoder, and output encoder. Finally, MorphNet has one decoder to predict the morph tags by conditioning on three encoder embeddings.

2.3 Part-of-Speech Tagging

It is challenging to describe parts of speech. It is a somewhat vague term. Schachter & Shopen (2018) defined the term *part of speech* distributionally; it means that words that have the same part-of-speech category have similar syntactic functions in the sentence. Exceptions can be presented with the substitution test, which controls the grammaticality of the sentence when the selected word substituted for each word belongs to the same part of speech category. Besides syntactic distribution, Hengeveld (1992) highlighted the importance of semantic roles of a word in the sentence. Semantically defined categories can add power to distributionally defined ones in cross-linguistics studies.

Although there are no precisely defined universal part of speech tag sets that capture every associated characteristic in all languages, a smaller version of a universal tag set is proposed for practical purposes. There are 45 tags, including punctuation, provided in Penn Treebank for English to code syntactic functions of the word (Marcus et al., 1993). Moreover, there are also cross-lingual, more finer-grainer tagsets are available; for instance, Universal Dependency (UD) datasets have 17 lexical tags which only capture core syntactic categories in all languages.

Part-of-speech (POS) tagging is a kind of sequence labeling problem. In this task, each word in an input sentence is labeled with a POS tag. Early examples of research into POS tagging were influenced by Harris' distributional hypothesis (Harris, 1954), which states that 'linguistic items with similar distributions have similar meanings'. From this point of view, the POS tagging problem can be seen as clustering words based on the distributional properties of syntax.

To date, in the literature, there are two conventional methods suggested to handle sequence labeling problem in the domain of POS tagging: generative and discriminative methods. Hidden Markov models (HMM) are based on a generative approach, and maximum entropy Markov models (MEMM) and recurrent neural networks (RNN) are based on the discriminative approach.

An HMM is based on the Markov chain, which describes the probabilities of a sequence of possible events where the probability of the next event only depends on the current event. However, in the task of POS tagging, events in the input sequence, i.e., POS tags, are not observable, they are hidden. For that reason, HMM models incorporated a new component, which is called as emission probability, into existing first-order Markov chain models. Emission probability matrix stores each of the probability of a token being generated from all possible POS tags, and is estimated from a labeled corpus. Thus far, several versions of HMM-based tagger Church (1989); DeRose (1988); Schütze & Singer (1994); Brants (2000); Tseng et al. (2005) are introduced to deal with problems such as unknown words.

An HMM tagger is conditioned on the observations and previous hidden states. Furthermore, it is not easy to incorporate arbitrary features such as tag features, morphological features, and multiword features. A MEMM tagger (Ratnaparkhi, 1996) predicts the best tag sequence given an observation, previous tags, and handcrafted features. Toutanova et al. (2003) presents the bidirectional version of MEMM tagger. Lafferty et al. (2001) reported the observational bias problem resided in MEMM and introduced a structured prediction technique called conditional random fields (CRF).

In recent years, researchers have investigated a variety of neural-based approaches to the POS tagging problem (Gillick et al., 2015; Ling et al., 2015; Plank et al., 2016). Especially, RNN based methods have been shown great success in sequence prediction tasks. Gillick et al. (2015) described an LSTM-based model, which they call Byte-to-Span (BTS). The proposed model is language-agnostic; it means that it does not have additional parameters for each distinct language. Ling et al. (2015) introduced BI-LSTM-based model to learn word representations by composing characters. The final word representations fed into a bidirectional LSTM model for POS tagging. Plank et al. (2016) introduced auxiliary loss function to account for rarely occurring words. This model gained state-of-art results, especially in languages that have rich morphology.

2.4 Dependency Parsing

Dependency grammar is a family of grammar formalism that describes the internal structure of the sentence in terms of directed relations between words in the sentence. There are no constituents as in Penn Treebank (Marcus et al., 1993). Dependency grammar formalism encodes the grammatical structure of sentence through the instrument of directed graph-like structure, which has a set of nodes correspond to lexical items in the sentence, and a set of labeled binary relations correspond to grammatical functions between "head" and "dependent" words. A notion of head exists in other grammar formalisms, such as phrase structure grammar (Gazdar, 1982) and head-driven phrase structure grammar (Pollard & Sag, 1994). In phrase structure grammar, within constituents, the head of the constituent governs arguments and adjuncts, but these relationships are not made explicit. Fortunately, dependency grammar is explicitly encoded head-dependent relationships without the need to require word-order information in contrast to phrase structure grammar.

In recent years, a considerable amount of effort has been devoted to the annotation of dependency treebanks especially for morphologically rich languages such as Prague dependency treebank (Hajičová, 1998), Basque dependency treebank (Aduriz et al., 2003), METU-Sabancı Turkish Dependency Treebank (Oflazer et al., 2003). We used Universal Dependency (UD) (Nivre et al., 2016) framework for grammar representation.

Dependency parsing is the task of automatically analyzing the dependency structure of sentences. In other words, dependency parsing is a mapping problem; it takes input sentence S and generates graph-like dependency structure G. There are two main approaches to dependency parsing problem: data-driven and grammar-based approaches. Data-driven approaches heavily make use of machine learning; however, grammar-based approaches are based on formal grammar. Most of the data-driven parsers either transition-based (Kudo & Matsumoto, 2002; Yamada & Matsumoto, 2003) and graph-based (Eisner, 1996; McDonald et al., 2005). Context-free (Hays, 1964; Gaifman, 1965) and constraint-based (Maruyama, 1990) parsers are the main approaches to grammar-based parsers.

In the last few decades, there has been a surge of interest in supervised methods for data-driven dependency parsing. In literature, unsupervised dependency parsing has gained little attention. Most-well known examples are (Yuret, 1998; Klein & Manning, 2005; Smith, 2006).

When dealing with morphologically rich languages, e.g., Turkish and Hungarian, morphological information needs to be integrated. Eryiğit et al. (2008) proposed a probabilistic parsing method for Turkish using the gold morphological analysis and POS tags provided by Oflazer et al. (2003). Eryiğit et al. (2008) shows the effect of morphology on syntactic parsing in Turkish.

In recent years, neural network-based models have been becoming a common trend in dependency parsing as well as other tasks in NLP (Chen & Manning, 2014; Dyer et al., 2015; Kiperwasser & Goldberg, 2016).

Chen & Manning (2014) introduced a neural transition-based dependency parsing model. Chen & Manning (2014) pointed out that manually designed or automatically captured features bring about sparsity problem since there is no adequate data to estimate these feature parameters correctly. The proposed (Chen & Manning, 2014) method was faster and more accurate (about 2% improvement) than existing works.

Dyer et al. (2015) presented their stack LSTM-based dependency parser, where the parser's state represented by fixed-dimensional continuous vectors. Additionally, Dyer et al. (2015) enable compositional representation of complex phrases using Recursive neural networks.

Kiperwasser & Goldberg (2016) presented a dependency parsing model in which uses two layered bidirectional LSTM (BI-LSTM) to encode sequences of tokens. Kiperwasser & Goldberg (2016) tested BI-LSTM based oracle on transition-based parsing. The resulting parser achieves comparable results with current state-of-the-art methods on the English (PTB) and Chinese (CTB) datasets.

Joint learning of morphology, word categories, and syntax becomes more widespread with the aid of end-to-end neural systems (Nguyen et al., 2017; Nguyen & Verspoor, 2018; L. Yang et al., 2017; Hashimoto et al., 2017).

Nguyen et al. (2017); Nguyen & Verspoor (2018) introduced and improved neural network-based joint model for POS tagging and graph-based dependency parsing. A similar dependency architecture is used in (Kiperwasser & Goldberg, 2016), but the character-based word embedding is used instead of word embeddings.

Hashimoto et al. (2017) introduce a deep multi-task learning model, which is based on a hierarchical parameter sharing mechanism, to learn several NLP tasks simultaneously. Five different NLP tasks are handled in separate levels: word level, syntactic level, and semantic level. Hashimoto et al. (2017) show that introducing supervision at word level can improve the overall performance of syntactic and semantic level tasks.

CHAPTER 3

MODEL

The section below describes the outline of the joint model for morphological segmentation, morpheme tagging, part-of-speech tagging, and dependency parsing.

3.1 Neural Building Blocks

The structure and functions of neural building blocks, which are used in models for sequence modeling, will be explained in the following section.

3.1.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) (Rumelhart et al., 1986) are a class of artificial neural networks designed to handle sequential data. The primary difference between RNNs and feedforward neural networks lies in shared parameters that enable RNNs to model samples of various lengths and to generalize across them. In other words, RNNs have shared parameters for each input feature, and thus RNNs can capture information about the dependencies in the sequential input along the temporal direction. The modern RNN types have been developed to deal with shortcomings of earlier versions such as Elman Networks (Elman, 1990).

RNNs can be built in many ways, but in all forms have a function involving recurrence. The mathematical notation of recurrent function that computes the hidden state of the network at a given timestep t for input sequence can be defined as a function f as follows:

$$h_t = f(h_{t-1}, x_t; \theta) \tag{3.1}$$

where X is an input sequence $(x_1, x_2, .., x_t, ...x_T)$ with length T, h_t represents the state information at time t, and θ refers to parameters set.

Typically, there is also another affine layer to project hidden information to outputs to make predictions. It is also possible to extend simple RNN with a stacked version by adding multiple recurrent hidden states on top of each other.

The full RNN model can be formally defined as:

$$\begin{aligned} h_t^l &= f(W_x h_t^{l-1} + W_h h_{t-1} + b_h^l) \\ o_t &= g(W_o h_t^l + b_o^l) \end{aligned} \tag{3.2}$$

where $W_x \in \mathbb{R}^{d_h, d_x}$ weight matrix for input $h_t^{l-1} \in \mathbb{R}^{d_h}$, $W_h \in \mathbb{R}^{d_h, d_h}$ weight matrix for $h_{t-1}^l \in \mathbb{R}^{d_h}$, and $b_h^l \in \mathbb{R}^{d_h}$ is bias term. The sum of these values is passed into suitable activation to compute the hidden state $h_t^l \in \mathbb{R}^{d_h}$. $W_o \in \mathbb{R}^{d_o, d_h}$ E dim and $b_o \in \mathbb{R}^{d_o}$ E dim are output layer parameters, and g is the output layer activation function. Typically, the softmax activation function is used as a g to compute normalized probability distribution.

Although RNNs theoretically can manage to encode long sequences of input, practically, it is not easy to learn long-term dependencies using simple RNNs. Bengio et al. (1994) have first identified that during the training phase in the backpropagation step gradients become close to zero due to repeated multiplications of the Jacobians at every timestep. Therefore parameter weights would not be updated, so the neural network does not learn. There is also another problem caused by the gradient explosion. Pascanu et al. (2013) proposed a simple way called a gradient clipping to resolve this issue.

3.1.2 Bidirectional RNNs

Simple RNNs process input sequence left-to-right manner, hence the hidden state of RNN at time t only encodes the information up to input x_t, i.e., the left context of the input. However, in some applications such as machine translation (Bahdanau et al., 2014), speech recognition (Graves et al., 2013) and, hand-writing recognition (Graves & Schmidhuber, 2009) requires both right and left context at each timestep. Bidirectional RNNs (Schuster & Paliwal, 1997) were designed to meet these needs.

Two separate RNNs are trained in the forward and backward directions of the input sequence and at each time step forward and backward outputs combined in some way to represent the state of the network at a given timestep t. Even though the conventional way of backward and forward context combination method is concatenation operation, there are some alternative methods such as element-wise addition and multiplication.

3.1.3 Long Short Term Memory

As discussed above, it is clear that RNNs appear to have difficulties while learning long-range dependencies. Because of the vanishing gradients problem in RNNs for the longer sequences, Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) networks, which learn to carry only the relevant hidden information (memory) to pass through the network and learn to forget the irrelevant bits, are preferred for longer sequences of dependencies. Thus far, several studies have confirmed the effectiveness of LSTMs in different areas of NLP, such as syntactic parsing (Vinyals et al., 2015), machine translation (Cho et al., 2014; Schuster & Paliwal, 1997), and language modeling (Graves, 2013). In this study, LSTM architecture proposed by (Zaremba et al., 2014) is adopted.

$$
\begin{aligned}
&\text{LSTM} : x, h_{t-1}, c_{t-1} \rightarrow h_t, c_t \\
&\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} W \begin{pmatrix} x \\ h_{t-1} \end{pmatrix} \\
&c_t = f \odot c_{t-1} + i \odot g \\
&h_t = o \odot \tanh(c_t)
\end{aligned}
\tag{3.3}
$$

where x , c, and h denotes input embedding, cell state, hidden state, respectively. sigm and tanh are element-wise operations. W is a $\mathbb{R}^{4n,2n}$ dimension weight matrix. The input modulation gate learns to neglect current input, the forget gate learns to determine how much of the previous hidden state should be kept. The forget gate could be sufficient to alleviate the risk of vanishing gradient problem in most cases.

3.1.4 Attention Mechanism

The term 'attention' has been used to describe the behavioral and cognitive process of selectively focusing on a discrete part of the information rather than the whole. There is no simple meaning of attention in psychology and is little consensus about what attention means.

According to James (1890):

> *Everyone knows what attention is. It is the taking possession by the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects or trains [p. 404] of thought. Focalization, concentration, of consciousness are of its essence. It implies withdrawal from some things in order to deal effectively with others, and is a condition which has a real opposite in the confused, dazed, scatterbrained state which in French is called distraction, and Zerstreutheit in German.*

Historically, attention mechanisms used in deep learning are loosely based on biologically inspired human visual attention mechanisms. The early investigations of attention mechanisms in NLP were made in the context of machine translation (Bahdanau et al., 2014). After that, attention-based neural network architectures have been shown great success in document classification (Z. Yang et al., 2016), semantic parsing (Dong & Lapata, 2016; Finegan-Dollak et al., 2018), and language modeling (Devlin et al., 2019).

Chaudhari et al. (2019) reviewed the neural architectures using attention mechanism, and provide the taxonomy of several attention mechanisms. For instance, document classification and machine translation models require different attention architectures. The former has an input sequence but not the output sequence; however, the latter has both input and output sequence. The multi-level inner attention type is proposed for the document classification task (Z. Yang et al., 2016), and mostly distinctive-soft attention type is proposed for machine learning tasks (Bahdanau et al., 2014).

Having defined what is meant by attention mechanism, let us now consider to define the formal definition of it. Due to being the predecessor of the modern attention mechanism, the attention architecture introduced by Bahdanau et al. (2014) will be briefly explained.

Recurrent neural networks based encoder-decoder architectures (Cho et al., 2014; Sutskever et al., 2014) have been used to model machine translation problems. Generally, the encoder reads the input sequence and produces the summary of input, which is called as context. Then decoder initializes its hidden vector using context information and generates the output. However, traditional encoder-decoder models suffer from two well-known deficiencies: fixed-sized representations and word alignment problem. First, RNN has to encode all input information into fixed-size representation. For instance, using a fixed-sized representation encoder must encode all the semantic aspects of a long sentence. Secondly, there is no alignment between the input and output sequences. Bahdanau et al. (2014) came up with an ingenious idea of attention. In the first phase of the process, the encoder takes all input data, then produces feature vector representations for each item in the input sequence. In the second phase of the process, at each decoding position j, attention network learns soft probabilities α_{ij} to compute context vector c_j by weighted averaging of all hidden representations of the input sequence. The attention model proposed by Bahdanau et al. (2014) can be formally defined as:

$$\begin{aligned} e_{ji} &= a(h_i^{in}, h_j^{out}) \\ \alpha_{ij} &= \frac{e_{ji}}{\sum_i e_{ji}} \\ c_j &= \sum_i \alpha_{ji} h_i^{in} \end{aligned} \tag{3.4}$$

where h_i^{in} is encoder hidden state at time i, h_j^{out} is decoder hidden state at the time j, and $a(h_i^{in}, h_j^{out})$ is alignment function which outputs a scalar score between h_j^{out} and h_i^{in} per input position i. e_{ji} is score between h_j^{out} and h_i^{in}, α_{it} is soft attention score, and c_j refers to weighted average of all hidden representations of input sequence called context.

3.2 The Proposed Model: Joint Morpheme Segmentation, Morpheme Tagging, POS Tagging, and Dependency Parsing

The following part of this section describes the architecture of the joint model. The proposed model handles the four related task-morpheme segmentation, morpheme tagging, pos tagging, and dependency tagging-in a joint manner. The model proposed by Nguyen & Verspoor (2018) and Kiperwasser & Goldberg (2016) provides the base model. It can be seen that this study is an extension of its predecessors. Morphological information, morpheme, and tag information are incorporated into the base model.

The base model, which is proposed by Nguyen & Verspoor (2018), has three layers. In the first layer, words are represented by their distributed word vectors are taken from pre-trained models, and their character-level representations are captured by BiLSTM. In the second layer, word vectors are fed into the BiLSTM model, then output representation of BiLSTM is passed to a multi-layer perceptron with one hidden layer to predict part-of-speech tags of words in the given context. In the third layer, pos tag representations of words and word vectors representations are fed into another BiLSTM, then created latent feature vectors

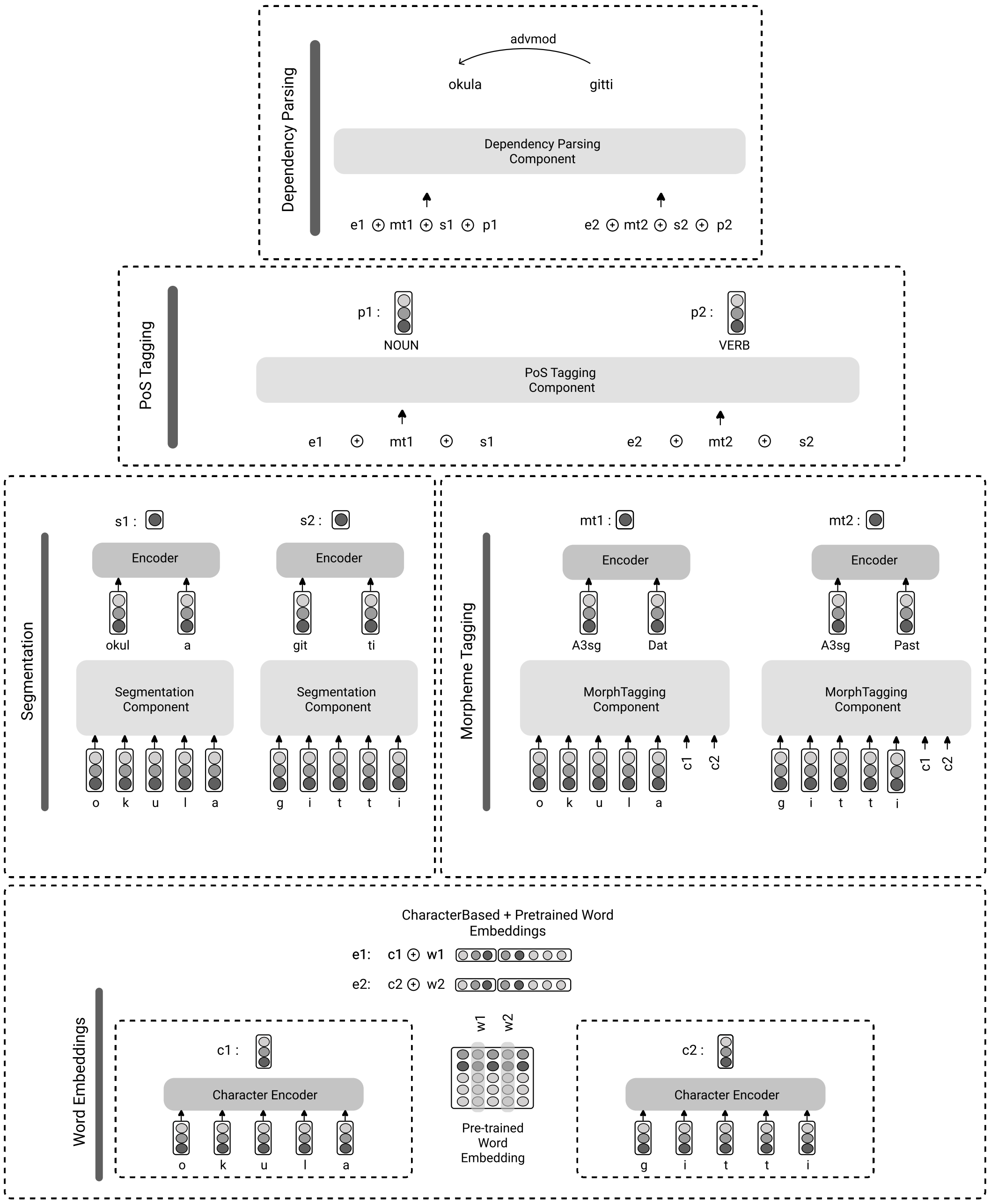

Figure 3.1: The overview of the full neural architecture that performs joint morpheme segmentation, morpheme tagging, PoS tagging, and dependency parsing.

are passed into multi-layer perceptron to predict dependency relations with their labels.

The proposed joint model extends the previous studies (Nguyen & Verspoor, 2018) by adding two extra components: morphological segmentation and morphological tagging model. The high-level sketch of the joint model is illustrated in Figure 3.1.

3.2.1 Morphological Segmentation

3.2.1.1 Input/Output Representations

The morphological segmentation component takes sequence characters of a word as an input, then produces sequences of binaries that corresponding to morpheme splitting points. Each word is represented as $w_i = [w_{i1}, ..., w_{ij}, ..., w_{iL_i}]$, $w_{ij} \in \mathbb{A}$ where w_i, w_{ij}, L_i, and $\mathbb{A}$ are ith word of sentence, jth character of word w_i, the length of word w_i, and the set of alphanumeric characters, respectively. The output is represented as $y_i = [y_{i1}, ..., y_{ij}, ..., y_{iL_i}]$, $y_{ij} \in \{0, 1\}$. Each item y_{ij} in output y_i can be 0 or 1 which indicates the segmentation point.

3.2.1.2 Model

The first layer of the joint model consists of BiLSTM and MLP layers. Each character w_{ij} in w_i is mapped to C-dimensional character embedding $x_{ij} \in \mathbb{R}^C$. Firstly, a BiLSTM takes sequences character embeddings $x_{i,1:L_i} \in \mathbb{R}^{L_i,C}$ as an input to encode latent feature representations.

$$v_j^{char} = BiLSTM_{seg}(x_{i,1:L_i}, j) \tag{3.5}$$

where v_j^{char} is a latent feature representation of jth character of word.

Each latent feature representation is passed to multi-layer perceptron with a sigmoid activation function as given below:

$$\hat{y}_j = \text{sigmoid}(MLP_{seg}(v_j^{char})) \tag{3.6}$$

The sigmoid activation function scales the input and outputs continuous range from 0 to 1. When the final prediction $\hat{y}_j$ number is close to 1, it is highly probable that there exists a segmentation boundary after that character. The binary cross-entropy function is computed for each position in the word. The segmentation loss is computed by formula 3.7:

$$\mathscr{L}_{seg} = -\sum_{j}^{L_i} (y_{ij} \log(\hat{y}_j) + (1 - y_{ij}) \log(1 - \hat{y}_j)) \tag{3.7}$$

where $\mathscr{L}_{seg}$ is the segmentation loss.

The morphological segmentation model is illustrated in Figure 3.2.

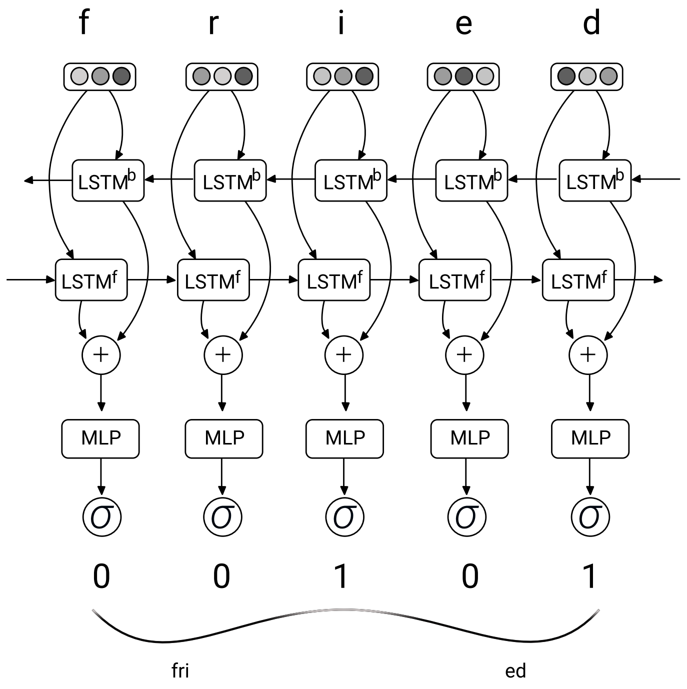

Figure 3.2: The architecture of the segmentation component.

3.2.2 Morphological Tagging

3.2.2.1 Input/Output Representations

The morphological tagging component takes sequence characters of a word as an input, then produces sequences of morphological tags as output. Morphological tagging and segmentation models have the same input representation. Output is represented as $f_i = [f_{i1}, ..., f_{ik}, ..., f_{iN_i}]$, $f_{ik} \in \mathbb{T}$ where f_i, f_{ik}, N_i, and $\mathbb{T}$ are morphological feature list of ith word, kth morphological feature of output, number of features in word w_i, and set of morphological features, respectively.

3.2.2.2 Model

There are three main parts of the morphological tagging component: word encoder, context encoder, and decoder. As previously described in the segmentation model, each character w_{ij} in w_i is mapped to C-dimensional character embedding $x_{ij} \in \mathbb{R}^C$. Firstly, a BiLSTM takes sequences character embeddings $x_{i,:} \in \mathbb{R}^{L_i,C}$ as an input to encode latent feature representations.

$$m_j^{char} = BiLSTM_{word\_encoder}(x_{i,1:L_i}, j) \tag{3.8}$$

where m_j^{char} is latent feature representation of jth character of word.

There is a many-to-one structural mismatch between sequences of characters and morpheme tags. For this reason, the classical encoder-decoder approach with fixed-size representation is not appropriate. The attention mechanism is added as an internal layer between word encoder and decoder in order to handle structural mismatches by reordering during the decoding phase. With the help of the attention mechanism, this model can learn the soft alignments between word input and morphological features. In this component, the attention network takes all latent feature representations produced by the encoder; then, at each decoding position k, the attention network learns soft probabilities α_{jk} by calculating the relation between each m_j^{char} and kth decoded token embedding t_k as follows:

$$\begin{aligned} score(m_j^{char}, t_k) &= v_1 \cdot tanh([U_1 m_j^{char}]; [W_1 t_k]) \\ e_{jk} &= score(m_j^{char}, t_k) \\ \alpha_{jk} &= \frac{e_{jk}}{\sum_j e_{jk}} \end{aligned} \tag{3.9}$$

where m_j^{char} is encoder hidden state at time j, t_k is decoder hidden state at the time k, and $score(m_j^{char}, t_k)$ is alignment function which outputs a scalar score between m_j^{char} and t_k each output position k. e_{jk} is score between m_j^{char} and t_k, and α_{jk} is soft attention score. And also, v_1, U_1, and W_1 are word attention network weights.

Once the weights are estimated, the total contribution of all characters to the current word is computed as the weighted sum of each character at time step k:

$$c_k = \Sigma_j \alpha_{jk} m_j^{char} \tag{3.10}$$

where c_k is the attended word characters of the current word that will be one of the inputs that will be fed into the decoder.

As for the word encoder, we define a second one-layer BiLSTM, where each context word is encoded as a vector, m_i^{word}:

$$m_j^{word} = BiLSTM_{context\_encoder}(x_{i:L_i}, j) \tag{3.11}$$

where m_j^{word} is latent feature representation of jth word of the sentence.

Another attention mechanism is added as an internal layer between context encoder and decoder to learn the soft alignments between sequences of words and morphological features. At each decoding position k, the context attention network learns soft probabilities α_{jk} by calculating the relation between each m_j^{word} and kth decoded token embedding t_k as follows:

$$\begin{aligned} score(m_j^{word}, t_k) &= v_2 \cdot tanh([U_2 m_j^{char}]; [W_2 t_k]) \\ e_{jk} &= score(m_j^{word}, t_k) \\ \alpha_{jk} &= \frac{e_{jk}}{\sum_j e_{jk}} \end{aligned} \tag{3.12}$$

where m_j^{word} is encoder hidden state at time j, t_k is decoder hidden state at the time k, and $score(m_j^{char}, t_k)$ is aligment function which outputs scalar score between m_j^{word} and t_k each output position k. e_{jk} is score between m_j^{word} and t_k, and α_{jk} is soft attention score. And also, v_2, U_2, and W_2 are context attention network weights.

Once the weights are estimated, the total contribution of all words in the sentence is computed as the weighted sum of each word at time step t:

$$wv_k = \Sigma_j \alpha_{kj} m_j^{word} \tag{3.13}$$

where wv_k is the attended context words that will be also another input to be fed into the decoder.

At each decoding position k, concatenation of context vector and previous decoded token embedding t_{k-1} are passed to the decoder. Then, at each decoding position k, the decoder output vectors are passed to MLP, then it produces output vector $v_k^{output} \in \mathbb{R}^{N_i}$. After that, the softmax activation is applied to output vector to get normalized probability distribution consisting of the size of $|\mathbb{T}|$ morphological features.

$$\begin{aligned} t_k &= c_k \circ wv_k \circ t_{k-1} \\ d_k^{token} &= LSTM_{decode}(t_{1:N_i}, k) \\ \hat{y_k} &= softmax(MLP_{mtag}(d_k^{token})) \end{aligned} \tag{3.14}$$

where t_k, N_i, and d_k^{token} are concatenated vector, number of features in word w_i, and decoder output representation of a morpheme tag, respectively.

Finally, categorical cross-entropy loss is computed for each position in the decoder. The morphological tagging loss is computed by formula 3.15:

$$\mathscr{L}_{mtag} = -\sum_{k}^{\mathbb{T}} f_{ik} \log(\hat{y_k}) \tag{3.15}$$

where $\mathscr{L}_{tag}$ is the morpheme tagging loss.

The morphological tagging model is illustrated in Figure 3.3.

3.2.3 Word Vector Representation

The following is a brief description of word vectors representation that will be used in POS tagging and dependency parsing components. Each sentence s is represented as $s = [w_1, ..., w_i, ..., w_n]$, where w_i corresponds to ith word of sentence s. For the POS tagging and dependency tasks, each word w_i is in s is mapped to C-dimensional word vector $x_i \in \mathbb{R}^C$. Each word vector x_i carry four different pieces of information: pre-trained word embedding $x_{w_i}^{(w)}$, character-level word embedding $x_{w_i}^{(c)}$, morpheme embedding, and morpheme tag embedding $x_{w_i}^{(m)}$.

$$x_i = x_{w_i}^{(w)} \circ x_{w_i}^{(c)} \circ x_{w_i}^{(m)} \circ x_{w_i}^{(mt)} \tag{3.16}$$

Pre-trained word embeddings are taken from word2vec (Mikolov et al., 2013). Character-level

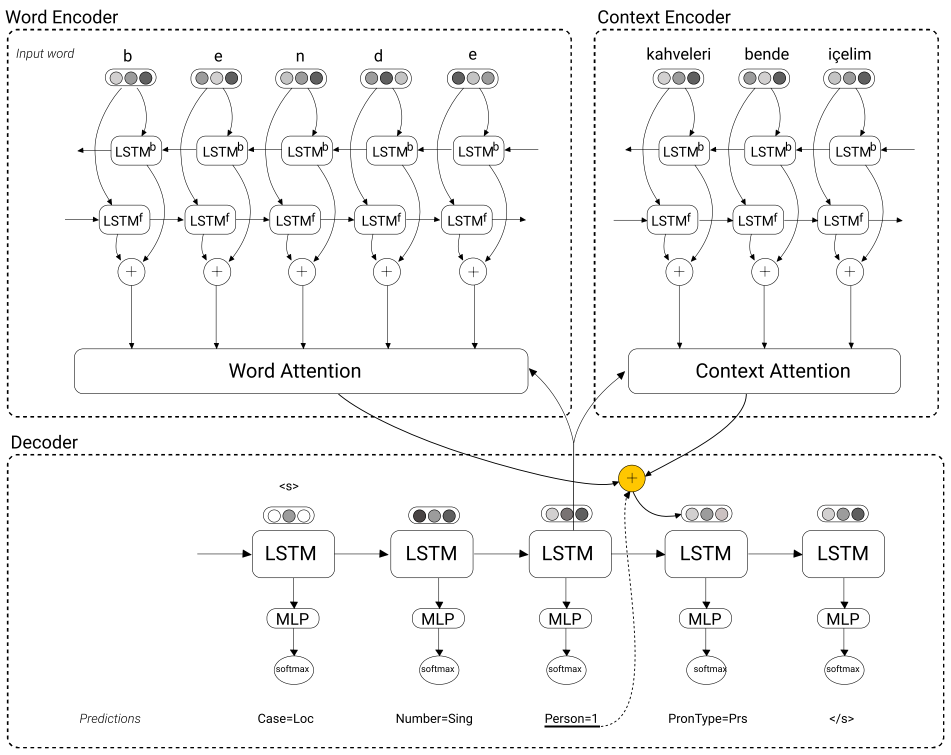

Figure 3.3: The encoder-decoder architecture of the morphological tagging component.

word embeddings are learned using the BiLSTM model. As previously stated in morphological segmentation part of this study, each word is represented as $w_i = [w_{i1}, ..., w_{ij}, ..., w_{iL_i}]$, $w_{ij} \in \mathbb{A}$ where w_i, w_{ij}, L_i, and $\mathbb{A}$ are ith word of sentence, jth character of word w_i, the length of word w_i, and the set of alphanumeric characters, respectively. Each character w_{ij} in w_i is mapped to C-dimensional character embedding $x_{ij} \in \mathbb{R}^C$.

For the character-level word embedding generation step, BiLSTM takes sequences of character embeddings $x_{i,1:L_i}$ of the word w_i. Then, the hidden representation of the last input time step is used as character-level word embedding.

$$x_{w_i}^{(c)} = BiLSTM_{char\_level\_embed}(x_{i,1:L_i}) \tag{3.17}$$

where $x_{w_i}^{(c)}$ is charecter-level feature representation of ith word of sentence.

The morph2vec model (Üstün et al., 2018) is re-implemented to learn morpheme embeddings. The morph2vec returns the sequence of morpheme embeddings for each word w_i. Moreover, a BiLSTM read these morpheme embeddings to produce single morpheme-level word representation $x_{w_i}^{(m)}$.

$$x_{w_i}^{(m)} = BiLSTM_{morph\_level\_embed}(x_i^m) \tag{3.18}$$

where $x_{w_i}^{(c)}$ is morpheme-level feature representation of ith word of sentence and x_i^m is the sequences of morpheme embeddings of w_i.

Finally, another a BiLSTM read these morpheme tag embeddings to produce single morpheme-tag word representation $x_{w_i}^{(mt)}$.

$$x_{w_i}^{(mt)} = BiLSTM_{morph\_tag\_embed}(x_i^{mt}) \tag{3.19}$$

3.2.4 POS Tagging

3.2.4.1 Input/Output Representations

The POS tagging component takes the sequences of word vector representations $x_s = [x_1, .., x_i, ..., x_{L_i}]$ as an input, then produces sequences of POS tags as output. Output representations is as $y_p = [y_{p_1}, ..., y_{p_i}, ..., y_{p_n}]$, $y_{p_i} \in \mathbb{P}$ where y_{p_k} is the gold POS tag of ith word, and $\mathbb{P}$ refers the set of POS tags.

3.2.4.2 Model

The POS tagging component consists of two layers: BiLSTM encoder and MLP-based classifier. BiLSTM reads the word vector representations, which are mentioned in the previous section, to encode latent feature representations.

$$v_i^{pos} = BiLSTM_{pos}(x_s) \tag{3.20}$$

where x_s denotes the sequence of word vectors representation of sentence s. v_i^{pos} is the POS representation of w_i.

At each time step, BILSTM feeds the output vectors into MLP with a softmax activation function to estimate part of speech tag of the word.

$$\hat{y} = softmax(MLPpos(v_i^{pos})) \tag{3.21}$$

where $\hat{y}$ is normalized probability distribution.

Categorical cross-entropy loss is used for training.

$$\mathscr{L}_{pos} = -\sum_{I}^{\mathbb{P}} y_{p_i} \log(v_i^{pos}) \tag{3.22}$$

where $\mathscr{L}_{pos}$ is the POS tagging loss.

3.2.5 Dependency Parsing

3.2.5.1 Input/Output Representations

The dependency parsing component takes word vector representation x_i for each word w_i in a sentence $s = [w_1, ..., w_i, ..., w_n]$. Each word vector x_i carry five different pieces of information:

pre-trained word embedding $x_{w_i}^{(w)}$, character-level word embedding $x_{w_i}^{(c)}$, morpheme embedding $x_{w_i}^{(m)}$, morpheme tag embedding $x_{w_i}^{(mt)}$,and POS tag embedding $x_{w_i}^{p}$ from POS tagging component .

$$x_i = x_{w_i}^{(w)} \circ x_{w_i}^{(c)} \circ x_{w_i}^{(m)} \circ x_{w_i}^{(mt)} \circ x_{w_i}^{(p)} \tag{3.23}$$

Dependency parsing tasks can be seen as a structured learning problem. Output representation has two parts: unlabeled dependency graph and labeled dependency relations. The directed dependency graph g of sentence $s = [w_1, ..., w_i, ..., w_n]$ is defined as $g = [parent(w_1), ..., parent(w_i), ..., parent(w_n)]$ where $parent(w_i)$ is function returns the parent id of word w_i. The corresponding labels l for each arc in graph g is reprenseted as $l = [l_1, ..., l_i, ..., l_n]$, $l_n \in \mathbb{L}$ where l_i and $\mathbb{L}$ denote label of w_i and set of dependency labels.

3.2.5.2 Model

In the first step, BiLSTM reads the word vector representations of sentence $x_{1:n}$ to encode latent feature representations.

$$v_i = BiLSTM_{dep}(x_{1:n}, i) \tag{3.24}$$

where x_s denotes the sequence of word vectors representation of sentence s. v_i is the latent feature representation of w_i.

Like (Nguyen & Verspoor, 2018) and (Kiperwasser & Goldberg, 2016), arc-factored parsing method is used to dependency arcs (McDonald et al., 2005; Kübler et al., 2009). The MLP based pointer networks are used to score arcs. All paired combinations of the latent feature representations are passed to pointer networks.

$$score_{arc}^{(i,j)} = MLP_{arc}(v_i \circ v_j \circ (v_i * v_j) \circ |v_i - v_j|) \tag{3.25}$$

where $score_{arc}^{(i,j)}$ is a function to compute strength of arc between head and dependent. v_i and v_j correspond to a sample pair.

The structure of the directed graph is extracted from the score matrix using the decoder algorithm proposed by Eisner (1996).

$$score(s) = \underset{\hat{y} \in \mathscr{Y}(s)}{\operatorname{argmax}} \sum_{(h,d) \in \hat{y}} score_{arc}^{(h,d)} \tag{3.26}$$

where $\mathscr{Y}(s)$ corresponds to all possible parse trees for sentence s. h and d denote the head and dependent nodes.

There is another MLP with a softmax activation function to predict dependency labels. Output label between the head and dependent arc is computed as follows:

$$\hat{y} = softmax(MLP_{label}(v_h \circ v_d \circ (v_h * v_d) \circ |v_h - v_d|)) \tag{3.27}$$

where $\hat{y}$, v_h, and v_d denote the predicted label, latent feature representations of head, and latent feature representations of headdependent words.

The dependency parsing model is illustrated in Figure 3.4.

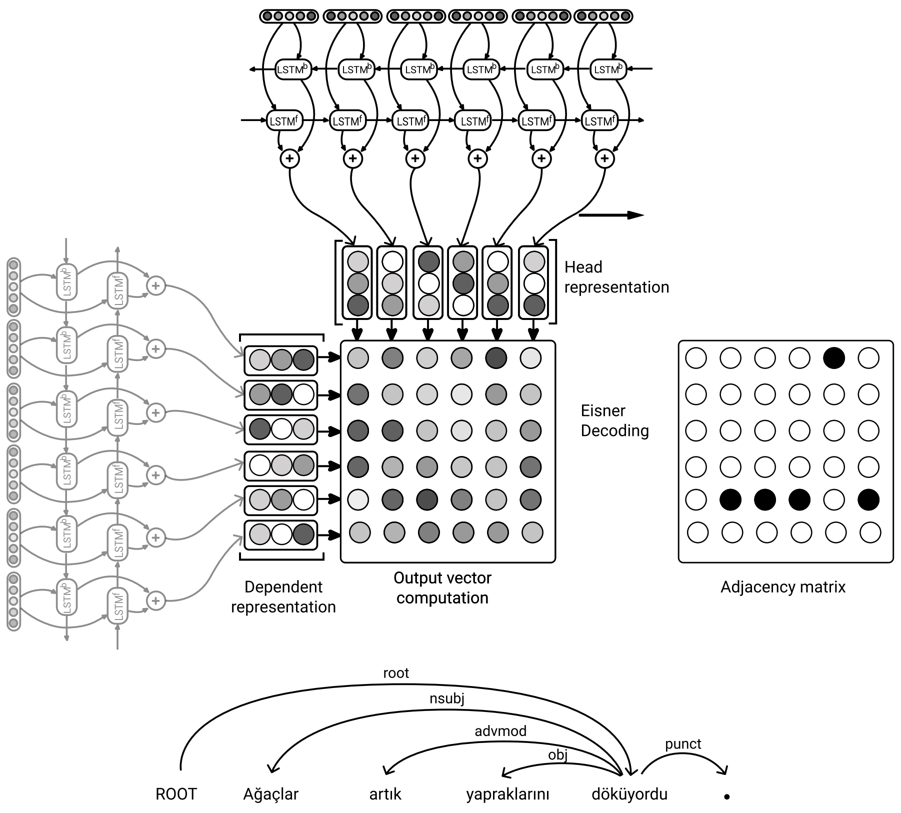

Figure 3.4: Pointer network representation that is used in the model.

3.2.6 Model Training

In the proposed joint learning framework, each component has its own error that contributes to the total loss of the model $\mathscr{L}$.

$$\mathscr{L} = \mathscr{L}_{seg} + \mathscr{L}_{mtag} + \mathscr{L}_{pos} + \mathscr{L}_{arc} + \mathscr{L}_{rel} \tag{3.28}$$

We use train and validation corpora to train the model. During training at each epoch, first the model is trained by Adaptive Moment Estimation (Adam) optimization algorithm (Kingma & Ba, 2014), and then per-token accuracy is calculated using the validation set. Every tenth epoch, all momentum values were cleared in the Adam Trainer. Here, we applied step based learning rate decay and early stopping.

CHAPTER 4

EXPERIMENTS

4.0.1 Data

We did the experiments on Turkish as a morphologically rich language. We used the UD Turkish Treebank for both training and evaluation. The dataset is a semi-automatic conversion of the IMST Treebank (Sulubacak et al., 2016), which is a reannotated version of the METU-Sabancı Turkish Treebank (Oflazer et al., 2003). All of the three treebanks share the same raw data that involves 5635 sentences collected from daily news reports and novels.

For the pre-trained word embeddings, we use pre-trained 200-dimensional word embeddings provided by CoNLL 2018 Shared Task. We also pre-train morph2vec (Üstün et al., 2018) to learn the morpheme embeddings. We obtain the gold segmentations from Zemberek (Akın & Akın, 2007) to train the segmentation component.

4.0.2 Implementation

We implemented the model using DyNet v2.0 (Neubig et al., 2017), a dynamic neural network library. For the morphological segmentation component, we split each token into characters and we represent each character with a 50-dimensional character embedding. Using the characters as input, we build a bi-directional LSTM with 50-dimensional unit size to encode the character context. We feed a multi-layer perceptron (MLP) with a sigmoid output that has an output size of 1 with the produced encoded characters to capture the segmentation boundaries.

Based on the segmentation probabilities obtained from the MLP, we created a morpheme list that makes the input word. We encode morphemes with 50-dimensional embeddings and we feed another Bi-directional LSTM with 50-dimensional units to encode each input morpheme. As mentioned before, we concatenate the 200-dimensional word-level word embeddings, 50-dimensional character-level word embeddings, and 100-dimensional morpheme-level word embeddings for the final word representation.

For the morpheme tagging component, we use a two-layered Bi-directional LSTM with 128-dimensional unit size to encode the character context. The decoder is also based on a two-layered Bi-directional LSTM with 128-dimensional unit size that is fed with the attended characters that are encoded by the encoder.

For the POS tagging component, we use a two-layered Bi-directional LSTM with 128-dimensional unit size to encode the word context. We feed the output of the word context into a MLP with a softmax activation function and 100-dimensional hidden unit size. Output size of the MLP is equal to the number of distinct POS tags in a given language. The softmax function gives the probability vector for the POS categories. We use the negative log-likelihood function as a loss function for the POS tagging component.

Finally, for the dependency parsing component, we use another two-layered Bi-directional LSTM with 128-dimensional unit size and a pointer network with 100-dimensional hidden unit size. We use again negative log-likelihood function for training.

We use dropout to penalize some random weights to zero. We observed significant improvement when we applied dropout with a rate of 0.3 after each LSTM units.

4.0.3 Results

We performed several experiments with the alternating components of the proposed model. All models are trained jointly with a single loss function. However, according to the desired joint tasks, we excluded either morpheme tagging or segmentation, or both tasks from the full model. Therefore, we observed how the lower levels that correspond to morpheme tagging and segmentation tasks affect the upper levels that refer to PoS tagging and dependency parsing.

We followed the standard evaluation procedure of CoNLL 2018 shared task defined for dependency parsing, morphological tagging and PoS tagging. For the PoS tagging, we evaluated our results with an accuracy score that is the percentage of the correctly tagged words. For the dependency parsing task, we present two different evaluation scores: labeled attachment score (LAS) and unlabeled attachment score (UAS). The attachment score is basically the percentage of the words that have the correct head and the dependency arc. LAS measures the percentage of the words that have the correct head with the correct dependency label, whereas UAS only measures the percentage of the words that have the correct head without considering the dependency labels. We evaluate the segmentation results with also accuracy, which is the percentage of the correctly segmented words. We evaluate the morpheme tagging with an accuracy measure, which is also the percentage of correctly tagged words. Here, we evaluate morpheme tagging based on all of the morpheme tags that each word involves. Therefore, we postulate that all the morpheme tags of the word to be correct in order to count the word as correctly tagged. It is defined as FEATS (universal morphological tags). Since the gold segmentations are not provided in the CoNLL datasets, we segmented the test set by Zemberek morphological segmentation tool Akın & Akın (2007) and evaluated based on the Zemberek results.

All the results obtained from the proposed models are given in Table 4.1. We compare our results with the joint PoS tagging and dependency parsing model by Nguyen & Verspoor (2018), which is the model that our joint model is built upon, and given as POS-DEP (Baseline) in the table. We obtained an UAS score of 70.42% and a LAS score of 62.71% from the baseline model. PoS tagging accuracy is the highest with 94.78% when all the components (morpheme tagging and morphological segmentation) are included as a full joint model. The same also applies to UAS, LAS, morpheme tagging accuracy and segmentation accuracy, which are the highest when all the components are adopted in the model. This shows that all the layers in the model contribute to each other during learning.

Table 4.1: Experimental results for different joint models.

	POS	UAS	LAS	FEATS	SEG
POS-DEP (Baseline)	92.16	70.42	62.71	-	-
MorphTag-POS-DEP	93.72	70.99	63.75	87.34	-
SEG-POS-DEP	94.51	70.99	62.62	-	98.90
SEG-MorphTag-POS-DEP	**94.78**	**71.00**	**63.92**	**87.59**	**98.97**

Since this is the first attempt that combines the four tasks in a single joint model, we compare the performance of each component separately with different models. The dependency parsing results are given in Table 4.2. The table shows the UAS and LAS scores of different dependency parsing models. We compare our results with the joint PoS tagging and dependency parsing model by Nguyen & Verspoor (2018), the winning of CoNLL 2018 shared task that incorporates deep contextualized word embeddings by Che et al. (2018), the pipeline system that performs tokenization, word segmentation, PoS tagging and dependency parsing by Qi et al. (2018), the tree-stack LSTM model by Kırnap et al. (2018), and the dependency parser that incorporates morphology-based word embeddings proposed by Özateş et al. (2018). The results show that our model makes a significant improvement on the baseline model of Nguyen & Verspoor (2018). Our UAS score is 3% higher and the LAS score is almost 4% higher than the joint PoS tagging and dependency parsing model by Nguyen & Verspoor (2018). Our model outperforms most of the models that participated in CoNLL 2017 and CoNLL 2018. The only model that perform better than our model is the one proposed by Che et al. (2018) and Qi et al. (2018). Their models give an UAS score of 72.25% and 71.07%, LAS score of 66.44% and 64.42% respectively, whereas our model gives an UAS score of 71.05% and LAS score of 63.32%. Our UAS score is competitive to the deep contextualized model by Che et al. (2018). However, there is a 3% difference in the LAS scores of the two models. As the authors of the work also state, deep contextualized word embeddings (BERT, Devlin et al. (2019)) affect the parsing accuracy significantly. However, apart from the usage of contextualized word embeddings, our base model is similar to their model. Ours performs morpheme tagging and morphological segmentation additionally. Qi et al. (2018) present another prominent model competed in CoNLL shared task 2018. They were placed on the 2nd in the shared task with their UAS and LAS scores. Similar to our model, they also introduce a neural pipeline system that performs tokenization, segmentation, PoS tagging, and dependency parsing. Their architecture is similar to ours. They use biaffine classifier for PoS tagging similar to that of Dozat & Manning (2016).

The comparison of our PoS tagging results with other PoS tagging models is given in Table 4.3. We compare our model with Che et al. (2018), Qi et al. (2018), Özateş et al. (2018), Kırnap et al. (2018). Our model outperforms other models with an accuracy of 95.02%. Following our model, the best performing model is the one proposed by Che et al. (2018), which uses deep contextualized word embeddings.

The comparison of our morphological tagging results (FEATS) with other models is given in Table 4.4. We compare our model with Che et al. (2018), Qi et al. (2018), Özateş et al. (2018), and Kırnap et al. (2018). The best performing model is by Qi et al. (2018) with an accuracy of

Table 4.2: The comparison of the Turkish dependency parsing results with other models.

Model	UAS	LAS
Che et al. (2018)	**72.25**	**66.44**
Qi et al. (2018)	71.07	64.42
Nguyen & Verspoor (2018)	70.53	62.55
Kırnap et al. (2018)	65.93	58.75
Özateş et al. (2018)	57.53	50.33
SEG-MORPH-POS-DEP	71.00	63.92
MORPH-POS-DEP	70.99	63.75
POS-DEP	70.42	62.71
SEG-POS-DEP	70.99	62.62

89.43%. It is worth to mentioning that none of those models perform joint learning for morphological tagging. Our model is the only joint model that performs morphological tagging along with other syntactic tasks. The results are very competitive with the ones of the other models although they fall behind the other models.

The results for English, Czech and Hungarian languages are given in Table 4.5.

Table 4.3: The comparison of the Turkish PoS tagging results with other models.

Model	Accuracy (UPOS)
Che et al. (2018)	**94.78**
Qi et al. (2018)	93.20
Özateş et al. (2018)	91.64
Kırnap et al. (2018)	91.64
Nguyen & Verspoor (2018)	92.93
SEG-MORPH-POS-DEP	**94.78**
SEG-POS-DEP	94.51
MORPH-POS-DEP	93.72

Table 4.5: The results on other languages

Language	Model	UAS	LAS	UPOS	FEATS
English	MORPH-POS-DEP	84.97	80.76	94.36	88.70
English	Che et al. (2018)	86.79	84.57	95.64	94.60
English	Nguyen & Verspoor (2018)	83.37	80.63	94.44	94.60
Czech	MORPH-POS-DEP	89.71	86.27	98.69	81.16
Czech	Che et al. (2018)	93.44	91.68	99.05	92.40
Czech	Nguyen & Verspoor (2018)	88.66	86.06	98.64	92.40
Hungarian	MORPH-POS-DEP	76.74	70.03	92.47	64.09
Hungarian	Che et al. (2018)	87.21	82.66	96.43	88.09
Hungarian	Nguyen & Verspoor (2018)	75.72	69.37	91.97	88.09

4.0.4 Error Analysis

Here, we endeavour to characterize the errors made by the proposed model variations in greater detail. The experiments aim to classify the parsing errors with respect to the linguistic

Table 4.4: The comparison of the Turkish morphological tagging results (FEATS) with other models.

Model	FEATS
Qi et al. (2018)	**89.43**
Che et al. (2018)	86.23
Özateş et al. (2018)	86.15
Kırnap et al. (2018)	86.15
SEG-MORPH-POS-DEP	87.59
MORPH-POS-DEP	87.34

and structural properties of the dependency graphs. In all experiments, the four different joint models have been trained and tested with the standard split of Turkish IMST universal dependencies[1]. Error analysis is carried out using four linguistic and structural properties, namely, sentence length, projectivity, arc type, and part-of-speech.

4.0.4.1 Effect of Sentence Length

We first try to see the effect of sentence length on the robustness of the four model variations. For that purpose, test data is subdivided into equally sized sentence length intervals. Here, interval length is 10. The UAS and LAS scores according to different sentence lengths are given in Table 4.7 and Table 4.6 respectively. It can be clearly seen that the longer the sentences are, the lower the both scores are for all model variations. However, MorphTag-POS-DEP and SEG-MorphTag-POS-DEP models perform rather well in shorter sentences (0-20 length) with regards to both UAS and LAS. What is interesting about the scores is that models that have higher UAS and LAS scores in shorter sentences are worse in longer sentences (30-50). Parsing errors gradually increase as the sentence lengths increase. In literature, a similar outcome was also reported for other languages such as for English and Vietnamese (McDonald & Nivre, 2007; Van Nguyen & Nguyen, 2015).

4.0.4.2 Effect of Projectivity

An arc in a dependency tree is projective if there is a directed path from the headword x to all the words between the two endpoints of the arc (Kübler et al., 2009) and therefore there are no crossing edges in the dependency graph. Projectivity is less frequent in languages with more strict order such as English. However, projective trees can be seen more frequently in free order languages such as Czech and German. Table 4.8 shows some statistics about the test set. Turkish being a free order language, the number of projective dependency graphs is

[1] Turkish IMST universal dependencies : https://universaldependencies.org/treebanks/tr
_imst/

Table 4.6: UAS by Length.

	<=10	<=20	<=30	<=40	<=50
POS-DEP	77.73	67.52	63.88	63.18	60.63
SEG-POS-DEP	78.86	67.95	**64.04**	61.59	**61.54**
MorphTag-POS-DEP	**79.07**	68.29	63.17	62.05	59.28
SEG-MorphTag-POS-DEP	78.71	**68.57**	62.50	**65.45**	60.63
Number of Tokens	3875	3516	1952	440	221

Table 4.7: LAS by Length.

	<=10	<=20	<=30	<=40	<=50
POS-DEP	70.27	59.95	55.33	54.77	**55.20**
SEG-POS-DEP	71.46	60.18	57.63	53.18	54.75
MorphTag-POS-DEP	**71.92**	60.95	**55.94**	54.32	52.94
SEG-MorphTag-POS-DEP	71.48	**61.43**	55.79	**59.09**	52.49
Number of Tokens	3875	3516	1952	440	221

a lot more higher than the number of non-projective dependency graphs.

Table 4.9 gives the LAS scores according to the sentence lengths. The results show that as the complexity of sentences increases and thereby the lengths of the sentences, the probability of generating non-projective trees also increases. Therefore, the number of projective trees is higher for the sentence lengths smaller than 30, however the number of non-projective trees is higher for the sentence lengths larger than 40. For example, all the sentences that are longer than 50 in length are all built as non-projective dependency trees.

We analyze the labeled and unlabeled accuracy scores obtained from the projective and non-projective trees in testing. As given in Table 4.10, the MorphTag-POS-DEP model outperforms other models both for labeled and unlabeled dependency arcs in projective trees. However, for the non-projective trees, the SEG-MorphTag-POS-DEP model outperforms other models, as can be seen in Table 4.11.

Table 4.8: Statistics about test partition of Turkish IMST universal dependencies

	Tokens	Sentences
Projective	7928	855
Non-projective	2076	120
Total	10004	975

Table 4.9: Projective and non-projective tree percentages grouped by the sentence length.

	<=10	<=20	<=30	<=40	<=50
Percentage of projective trees	91%	82%	78%	40%	0%
Percentage of non-projective trees	19%	18%	22%	60%	100%
Number of tokens	3875	3516	1952	440	221

Table 4.11: Results for non-projective sentences.

	UAS	LAS
POS-DEP	60.69	53.81
SEG-POS-DEP	60.84	54.62
MorphTag-POS-DEP	60.45	54.05
SEG-MorphTag-POS-DEP	**62.43**	**55.39**

4.0.4.3 Effect of Linguistic Constructions

Finally, we also aim to analyze the syntactic properties of the dependency relations based on their PoS labels obtained from different models. Here, we analyze the results based on the dependent's PoS tag to see whether the dependent's PoS tag plays a role in the accuracy. As McDonald & Nivre (2007) proposed, we make use of coarse PoS categories instead of universal POS (UPOS) categories in order to observe the performance differences between the models easily for the sake of the loss of useful syntactic information. As for the coarse PoS categories, nouns include nouns and proper nouns, pronouns consist of pronouns and determiner, and verbs include verbs and auxiliary verbs.

UAS and LAS accuracies of the four models are given in Table 4.12 and Table 4.13 respectively for the coarse PoS categories. As can be seen from Table 4.12, there is a slight improvement of UAS in conjunctions (CONJ) and substantial improvement in verbs (VERB) and adpositions (ADP) in the SEG-MorphTag-POS-DEP model. The MorphTag-POS-DEP model captures the unlabeled arcs for the adjective (ADJ) and adverb (ADV) dependents better compared to the other models. The scores also show that there is a slight increase in the

Table 4.10: Results for projective sentences.

	UAS	LAS
POS-DEP	72.97	65.05
SEG-POS-DEP	73.65	65.98
MorphTag-POS-DEP	**73.75**	**66.30**
SEG-MorphTag-POS-DEP	73.25	66.16

Table 4.12: UAS by coarse POS categories.

Dependent POS	Total	MT	MT	MT	MT
		POS-DEP	SEG-POS	MorphTag	SEG-MorphTag
			-DEP	-POS-DEP	-POS-DEP
ADJ	939	72.52	72.63	**74.01**	72.31
ADP	360	76.11	81.39	78.89	**80.56**
ADV	345	**68.70**	65.22	**68.70**	66.09
CONJ	390	75.38	74.10	74.62	**75.64**
NOUN	3145	65.47	**66.83**	66.20	65.98
PRON	530	76.60	**77.17**	75.85	76.61
VERB	2121	72.09	71.28	72.99	**73.51**

unlabeled accuracy for the noun (NOUN) and pronoun (PRON) dependents in the SEG-POS-DEP model. The POS-DEP model does not outperform other models for none of the PoS categories. Only the UAS score for the adverb dependents is the same as the one obtained from the MorphTag-POS-DEP. The overall results show that each component in the model has an impact on the UAS scores for different PoS categories of the dependents, and the full joint model that includes all the components has a higher overall UAS accuracy for different dependent PoS categories.

As can be seen from Table 4.13, SEG-MorphTag-POS-DEP model tends to be more accurate in nouns (NOUN) and conjunctions (CONJ). With the addition of the morpheme tags, MorphTag-POS-DEP model has higher LAS accuracies for adjectives (ADJ), adverbs (ADV), and verbs (VERB). There is a considerable improvement of LAS in adpositions (ADP) after incorporating segmentation information in the joint SEG-POS-DEP model. However, SEG-POS-DEP model does not outperform other models for other PoS categories. The only PoS category that the POS-DEP outperforms other models is the pronouns (PRON), which can be a sign that pronouns are usually not inflected and the morph tags and segmentation do not contribute much for the pronoun category.

Table 4.13: LAS by coarse POS categories.

Dependent POS	Total	MT POS-DEP	MT SEG-POS -DEP	MT MorphTag -POS-DEP	MT SEG-MorphTag -POS-DEP
ADJ	939	57.51	59.74	**60.38**	60.06
ADP	360	73.61	**79.4**4	76.39	78.06
ADV	345	60.58	58.26	**61.74**	60.58
CONJ	390	69.49	66.92	66.41	**69.74**
NOUN	3145	53.90	55.86	55.74	**56.00**
PRON	530	**68.68**	67.73	67.36	67.36
VERB	2121	66.95	66.38	**68.18**	67.89

4.0.5 Pipeline Ablation

In order to measure the contribution of each component in the model, typically, that component is removed from the model. It is the so-called ablation analysis, and actually, we have done this kind of coarse version of ablation analysis in previous sections. There is also another version of ablation analysis that measures the influence of the individual components in the model by incrementally replacing the component with their gold annotations. This pipeline version of ablation analysis was studied by Qi et al. (2018). Using this approach, researchers have been able to decide whether an individual component is useful for higher-level tasks or not and estimate the limitations of their studies in perfect conditions. Moreover, the errors that flow through the network from different components are also filtered out to measure the actual effect of each component on the final scores.

The results obtained from the pipeline ablation analysis of the joint model are given in Table 4.14. The first row gives the results of the full joint model without gold annotations. In the second row (POS-MorphTag-GoldSEG-DEP), the morpheme segmentation component is replaced with the gold morphological segmentations rather than predicting them, and the rest of the components are left as they are. In the third row (POS-GoldMorphTag-SEG-DEP), the morphological tagging component is replaced with the gold morpheme tags. Finally, both morpheme segmentation and morphological tagging components are replaced with their gold annotations in the fourth row (POS-GoldMorphTag-GoldSEG-DEP). The results show that using the gold morpheme tags contributes to the dependency scores the most, however the highest improvement is obtained when both gold morphological segmentation and gold morpheme tags are incorporated in the model for the PoS tagging. Even if the individual components are replaced with the gold annotations incrementally from bottom tasks to the upper layers, there is still a huge gap between the ablation scores and the gold scores. The morphological tagging accuracy is 100% for the last two rows, because the gold morpheme tags have been used for these two ablation tests. Another natural finding of the ablation test is that when the gold morphological segmentations are incorporated in the model, the morpheme tagging accuracy can be also improved, which is an expected result.

Table 4.14: Pipeline ablation results.

	POS	UAS	LAS	UFeats
POS-MorphTag-SEG-DEP	94.78	71.00	63.92	87.59
POS-MorphTag-GoldSEG-DEP	94.90	70.93	64.05	87.81
POS-GoldMorphTag-SEG-DEP	97.30	**72.34**	**66.68**	100
POS-GoldMorphTag-GoldSEG-DEP	**97.36**	71.97	65.94	100

CHAPTER 5

CONCLUSION

The main goal of the current study was to demonstrate that the four tasks–morphological segmentation, morpheme tagging, POS tagging, and dependency parsing– which related to morphology and syntax can be learned efficiently in a single unified framework. The second aim of this study was to investigate the effects of morpheme and character representations to cope with the sparsity problem, which is introduced especially by agglutinative languages such as Turkish due to their complex morphology.

The results of this investigation show that morphological information such as morpheme tags and morphemes of the word improve both dependency parsing and PoS tagging results, which indicates that morphemic knowledge plays a vital role in syntactic learning of morphologically rich languages. The present study has been one of the first attempts to combine morphological segmentation, morpheme tagging, POS tagging, and dependency parsing in a single framework using deep neural networks. The results of this research support the idea that morphemic knowledge plays an essential role in syntactic learning of morphologically rich languages.

Current study did not include vertical information flow left-to-right or right-to-left manner. Further research could also be conducted to determine the effectiveness of vertical information flow. As a future goal in this work, the study should be repeated using deep contextualized word embeddings such as BERT (Devlin et al., 2019) or Elmo (Peters et al., 2018). Finally, further work could be conducted to determine the effectiveness of self-attention mechanism in the current architecture.

Bibliography

Aduriz, I., et al. (2003). Construction of a basque dependency treebank.

Akın, A. A., & Akın, M. D. (2007). Zemberek, an open source NLP framework for Turkic Languages. , *10*.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bengio, Y., Simard, P., Frasconi, P., et al. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157–166.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *TACL*, *5*, 135-146. Retrieved from `http://dblp.uni-trier.de/db/journals/tacl/tacl5.html#BojanowskiGJM17`

Brants, T. (2000, April). TnT – a statistical part-of-speech tagger. In *Sixth applied natural language processing conference* (pp. 224–231). Seattle, Washington, USA: Association for Computational Linguistics. Retrieved from `https://www.aclweb.org/anthology/A00-1031` doi: 10.3115/974147.974178

Can, B., & Manandhar, S. (2013, October). Dirichlet processes for joint learning of morphology and PoS tags. In *Proceedings of the sixth international joint conference on natural language processing* (pp. 1087–1091). Nagoya, Japan: Asian Federation of Natural Language Processing. Retrieved from `https://www.aclweb.org/anthology/I13-1152`

Chaudhari, S., Polatkan, G., Ramanath, R., & Mithal, V. (2019). An attentive survey of attention models. *arXiv preprint arXiv:1904.02874*.

Che, W., Liu, Y., Wang, Y., Zheng, B., & Liu, T. (2018). Towards better ud parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the conll 2018 shared task: Multilingual parsing from raw text to universal dependencies* (pp. 55–64). Association for Computational Linguistics. Retrieved from `http://aclweb.org/anthology/K18-2005`

Chen, D., & Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 740–750).

Cheng, H., Fang, H., He, X., Gao, J., & Deng, L. (2016). Bi-directional attention with agreement for dependency parsing. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 2204–2214). Association for Computational Linguistics. Retrieved from `http://aclweb.org/anthology/D16-1238` doi: 10.18653/v1/D16-1238

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., &

Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Church, K. W. (1989). A stochastic parts program and noun phrase parser for unrestricted text. In *International conference on acoustics, speech, and signal processing,* (pp. 695–698).

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011, November). Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, *999888*, 2493–2537. Retrieved from `http://dl.acm.org/citation.cfm?id=2078183.2078186`

Cöltekin, C. (2010). A freely available morphological analyzer for turkish. In *Lrec* (Vol. 2, pp. 19–28).

Cotterell, R., & Heigold, G. (2017). Cross-lingual, character-level neural morphological tagging. *arXiv preprint arXiv:1708.09157*.

Creutz, M., & Lagus, K. (2002). Unsupervised discovery of morphemes. In *Proceedings of the acl-02 workshop on morphological and phonological learning-volume 6* (pp. 21–30).

Creutz, M., & Lagus, K. (2005a). Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of the international and interdisciplinary conference on adaptive knowledge representation and reasoning (akrr'05)* (Vol. 1, pp. 51–59).

Creutz, M., & Lagus, K. (2005b). *Unsupervised morpheme segmentation and morphology induction from text corpora using morfessor 1.0.* Helsinki University of Technology Helsinki.

Dayanik, E. (2018). Morphological Tagging and Lemmatization with Neural Components.

DeRose, S. J. (1988). Grammatical category disambiguation by statistical optimization. *Computational linguistics*, *14*(1), 31–39.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Naacl-hlt.*

Dong, L., & Lapata, M. (2016). Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280.*

Dozat, T., & Manning, C. D. (2016). Deep biaffine attention for neural dependency parsing. *CoRR*, *abs/1611.01734*. Retrieved from `http://arxiv.org/abs/1611.01734`

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., & Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075.*

Eisner, J. M. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on computational linguistics-volume 1* (pp. 340–345).

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179–211.

Eryiğit, G., Nivre, J., & Oflazer, K. (2008). Dependency parsing of turkish. *Computational Linguistics*, *34*(3), 357–389.

Finegan-Dollak, C., Kummerfeld, J. K., Zhang, L., Ramanathan, K., Sadasivam, S., Zhang, R., & Radev, D. (2018). Improving text-to-sql evaluation methodology. *arXiv preprint arXiv:1806.09029.*

Gaifman, H. (1965). Dependency systems and phrase-structure systems. *Information and control*, *8*(3), 304–337.

Gazdar, G. (1982). Phrase structure grammar. In *The nature of syntactic representation* (pp. 131–186). Springer.

Gillick, D., Brunk, C., Vinyals, O., & Subramanya, A. (2015). *Multilingual language processing from bytes.*

Göksel, A., & Kerslake, C. (2005). Turkish: A comprehensive grammar..

Goldsmith, J. (2001). Unsupervised learning of the morphology of a natural language. *Computational linguistics*, *27*(2), 153–198.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850.*

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 ieee international conference on acoustics, speech and signal processing* (pp. 6645–6649).

Graves, A., & Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems* (pp. 545–552).

Hajičová, E. (1998). Prague dependency treebank: From analytic to tectogrammatical annotations. *Proceedings of 2nd TST, Brno, Springer-Verlag Berlin Heidelberg New York*, 45–50.

Harris, Z. S. (1954). Distributional structure. *Word*, *10*(2-3), 146–162.

Harris, Z. S. (1955, apr). From Phoneme to Morpheme. *Language*, *31*(2), 190. Retrieved from `https://www.jstor.org/stable/411036?origin=crossref` doi: 10.2307/411036

Harris, Z. S. (1970). Morpheme boundaries within words: Report on a computer test. In *Papers in structural and transformational linguistics* (pp. 68–77). Springer.

Hashimoto, K., Xiong, C., Tsuruoka, Y., & Socher, R. (2017, September). A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 1923–1933). Copenhagen, Denmark: Association for Computational Linguistics. Retrieved from `https://www.aclweb.org/anthology/D17-1206` doi: 10.18653/v1/D17-1206

Hays, D. G. (1964). Dependency theory: A formalism and some observations. *Language*, *40*(4), 511–525.

Heigold, G., Neumann, G., & van Genabith, J. (2016). Neural morphological tagging from characters for morphologically rich languages. *arXiv preprint arXiv:1606.06640.*

Heigold, G., & Van Genabith, J. (2017). An Extensive Empirical Evaluation of Character-Based Morphological Tagging for 14 Languages. *Eacl*, *1*(2016), 505–513. Retrieved from http://www.aclweb.org/anthology/W17-4118{%}0Ahttp://arxiv.org/abs/1706.01723

Hengeveld, K. (1992). Parts of speech. In *Layered structure and reference in a functional perspective* (p. 29). John Benjamins Publishing Company. Retrieved from https://doi.org/10.1075/pbns.23.04hen doi: 10.1075/pbns.23.04hen

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

James, W. (1890). *The principles of psychology*. Dover Publications.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, *abs/1412.6980*. Retrieved from http://arxiv.org/abs/1412.6980

Kiperwasser, E., & Goldberg, Y. (2016). Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, *4*, 313–327.

Kırnap, Ö., Dayanık, E., & Yuret, D. (2018). Tree-stack lstm in transition based dependency parsing. In *Proceedings of the conll 2018 shared task: Multilingual parsing from raw text to universal dependencies* (pp. 124–132). Association for Computational Linguistics. Retrieved from http://aclweb.org/anthology/K18-2012

Klein, D., & Manning, C. D. (2005). *The unsupervised learning of natural language structure*. Stanford University Stanford, CA.

Kübler, S., McDonald, R., & Nivre, J. (2009). Dependency parsing. *Synthesis lectures on human language technologies*, *1*(1), 1–127.

Kudo, T., & Matsumoto, Y. (2002). Japanese dependency analysis using cascaded chunking. In *proceedings of the 6th conference on natural language learning-volume 20* (pp. 1–7).

Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Ling, W., Dyer, C., Black, A. W., Trancoso, I., Fermandez, R., Amir, S., ... Luís, T. (2015, September). Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1520–1530). Lisbon, Portugal: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/D15-1176 doi: 10.18653/v1/D15-1176

Malaviya, C., Gormley, M. R., & Neubig, G. (2018). Neural factor graph models for cross-lingual morphological tagging. *arXiv preprint arXiv:1805.04570*.

Marcus, M., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: The penn treebank.

Maruyama, H. (1990). Structural disambiguation with constraint propagation. In *Proceedings of the 28th annual meeting on association for computational linguistics* (pp. 31–38).

McDonald, R., & Nivre, J. (2007). Characterizing the errors of data-driven dependency parsers.

McDonald, R., Pereira, F., Ribarov, K., & Hajič, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on human language technology and empirical methods in natural language processing* (pp. 523–530).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Müller, T., Schmid, H., & Schütze, H. (2013). Efficient higher-order CRFs for morphological tagging. *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*(October), 322–332.

Narasimhan, K., Barzilay, R., & Jaakkola, T. (2015). An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics*, *3*, 157–167.

Neubig, G., Dyer, C., Goldberg, Y., Matthews, A., Ammar, W., Anastasopoulos, A., ... Yin, P. (2017). Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980.*

Nguyen, D. Q., Dras, M., & Johnson, M. (2017). A novel neural network model for joint pos tagging and graph-based dependency parsing. *arXiv preprint arXiv:1705.05952.*

Nguyen, D. Q., & Verspoor, K. (2018). An improved neural network model for joint pos tagging and dependency parsing. *arXiv preprint arXiv:1807.03955.*

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C. D., ... Zeman, D. (2016, May). Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the tenth international conference on language resources and evaluation (LREC'16)* (pp. 1659–1666). Portorož, Slovenia: European Language Resources Association (ELRA). Retrieved from `https://www.aclweb.org/anthology/L16-1262`

Oflazer, K., Say, B., Hakkani-Tür, D. Z., & Tür, G. (2003). Building a turkish treebank. In *Treebanks* (pp. 261–277). Springer.

Özateş, c. B., Özgür, A., Gungor, T., & Öztürk, B. (2018). A morphology-based representation model for lstm-based dependency parsing of agglutinative languages. In *Proceedings of the conll 2018 shared task: Multilingual parsing from raw text to universal dependencies* (pp. 238–247). Association for Computational Linguistics. Retrieved from `http://aclweb.org/anthology/K18-2024`

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning* (pp. 1310–1318).

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018, 02). Deep contextualized word representations.

Petrov, S., Das, D., & McDonald, R. (2012). A universal part-of-speech tagset. In *Proceedings of the 8th international conference on language resources and evaluation, lrec 2012.*

Plank, B., Søgaard, A., & Goldberg, Y. (2016). *Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss.*

Pollard, C., & Sag, I. A. (1994). *Head-driven phrase structure grammar*. University of Chicago Press.

Qi, P., Dozat, T., Zhang, Y., & Manning, C. D. (2018). Universal dependency parsing from scratch. In *Proceedings of the conll 2018 shared task: Multilingual parsing from raw text to universal dependencies* (pp. 160–170). Association for Computational Linguistics. Retrieved from `http://aclweb.org/anthology/K18-2016`

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Conference on empirical methods in natural language processing.*

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Sak, H., Güngör, T., & Saraçlar, M. (2008). Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In B. Nordström & A. Ranta (Eds.), *Advances in natural language processing* (pp. 417–427). Berlin, Heidelberg: Springer Berlin Heidelberg.

Schachter, P., & Shopen, T. (2018). *Parts-of-speech systems.*

Schone, P., & Jurafsky, D. (2001). Knowledge-free induction of inflectional morphologies. In *Second meeting of the north american chapter of the association for computational linguistics.*

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

Schütze, H., & Singer, Y. (1994). Part-of-speech tagging using a variable memory markov model. In *Proceedings of the 32nd annual meeting on association for computational linguistics* (pp. 181–187).

Smith, N. (2006). *Novel estimation methods for unsupervised discovery of latent structure in natural language text* (Unpublished doctoral dissertation). Johns Hopkins University.

Soricut, R., & Och, F. (2015). Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 conference of the north american chapter of the association for computational linguistics: Human language technologies* (pp. 1627–1637).

Sulubacak, U., Pamay, T., & Eryigit, G. (2016). IMST: A revisited Turkish dependency treebank. In *The first international conference on turkic computational linguistics-turcling* (Vol. 2016, pp. 1–6).

Sutskever, I., Vinyals, O., & Le, Q. (2014). Sequence to sequence learning with neural networks. *Advances in NIPS.*

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 conference of the north american chapter of the association for computational linguistics on human language technology-volume 1* (pp. 173–180).

Tseng, H., Jurafsky, D., & Manning, C. (2005). Morphological features help POS tagging of unknown words across language varieties. In *Proceedings of the fourth SIGHAN* *workshop on Chinese language processing.* Retrieved from `https://www.aclweb.org/anthology/I05-3005`

Üstün, A. (2017). *Probabilistic learning of turkish morphosemantics by latent syntax* (Unpublished master's thesis). Middle East Technical University, Ankara.

Üstün, A., & Can, B. (2016). Unsupervised morphological segmentation using neural word embeddings. In *International conference on statistical language and speech processing* (pp. 43–53).

Üstün, A., Kurfalı, M., & Can, B. (2018). Characters or morphemes: How to represent words? In *Proceedings of the third workshop on representation learning for nlp* (pp. 144–153).

Van Nguyen, K., & Nguyen, N. L.-T. (2015). Error analysis for vietnamese dependency parsing. In *2015 seventh international conference on knowledge and systems engineering (kse)* (pp. 79–84).

Vinyals, O., Kaiser, Ł., Koo, T., Petrov, S., Sutskever, I., & Hinton, G. (2015). Grammar as a foreign language. In *Advances in neural information processing systems* (pp. 2773–2781).

Yamada, H., & Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In *Proceedings of the eighth international conference on parsing technologies* (pp. 195–206).

Yang, L., Zhang, M., Liu, Y., Sun, M., Yu, N., & Fu, G. (2017). Joint pos tagging and dependence parsing with transition-based neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *26*(8), 1352–1358.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies* (pp. 1480–1489).

Yarowsky, D., & Wicentowski, R. (2000). Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th annual meeting on association for computational linguistics* (pp. 207–216).

Yuret, D. (1998). Discovery of linguistic relations using lexical attraction. *arXiv preprint cmp-lg/9805009.*

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329.*

Zielinski, A., & Simon, C. (2009). Morphisto –an open source morphological analyzer for german. In *Proceedings of the 2009 conference on finite-state methods and natural language processing: Post-proceedings of the 7th international workshop fsmnlp 2008* (p. 224–231). NLD: IOS Press.