Morphological Tagging and Lemmatization with Neural Components

by

Erenay Dayanık

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

June, 2018

Morphological Tagging and Lemmatization with Neural Components

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Erenay Dayanık

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assoc. Prof. Deniz Yuret

_____

Asst. Prof. Gülşen Cebiroğlu Eryiğit

_____

Assoc. Prof. Engin Erzin

Date: _____

ABSTRACT

I describe and evaluate MorphNet, a language-independent, end-to-end model that is designed to combine morphological analysis and disambiguation. Traditionally, analysis of morphologically complex languages has been performed in two stages: (i) A morphological analyzer based on finite-state transducers produces all possible morphological analyses of a word, (ii) A statistical disambiguation model picks the correct analysis based on the context for each word. MorphNet uses a sequence-to-sequence recurrent neural network to combine analysis and disambiguation. The model consists of three LSTM encoders to create embeddings of various input features and a two layer LSTM decoder to predict the correct morphological analysis. When MorphNet is trained with text labeled with correct morphological analyses, the model is able to achieve state-of-the art or comparable results in twenty-six different languages.

ÖZETÇE

Bu tezde MorphNet adını verdiğim, dilden bağımsız, uçtan-uca çalışan bir nöral model tarif ettim. MorphNet Biçimbilimsel Çözümleme ve Belirsizlik Giderme uygulamalarını eş zamanlı bir şekilde yapabilmek için tasarlanmıştır. Geleneksel olarak, biçimbilimsel olarak karmaşık dillerin analizi iki aşamalı olarak yapılmaktadır: (i) Sonlu durumlu dönüştürücüler tabanlı Biçimbilimsel Çözümleme uygulaması hedef sözcüğün olası tüm görülüş biçimlerini listeler, (ii) İstatistiksel Biçimbilimsel belirsizlik giderme uygulaması da kelimenin içinde bulunduğu bağlamı dikkate alarak doğru görülüş biçimini seçer. MorphNet'te diziden diziye nöral ağlar mimarisi kullanılarak Biçimbilimsel Çözümleme ve Belirsizlik giderme uygulamaları bir araya getirilmiştir. Modelde üç farklı Geri Dönüşümlü Yapay Sinir Ağı girdi kelimelerin çesitli özelliklerinin vektör uzayında kodlanmasında ve bir adet iki katmanlı Geri Dönüşümlü Yapay Sinir Ağı doğru biçimsel görülüşün üretilmesi için kullanılmaktadır. Bu tezde, doğru biçimbilimsel analizler ile etiketlenmiş data ile eğitildiğinde, MorphNet'in yirmi altı dilde geçmiş sistemlerden daha iyi veya kıyaslanabilir sonuçlar elde ettiği gösterilmiştir.

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor Deniz Yuret for his immense knowledge and support. I would like to thank Gülşen Cebiroğlu Eryiğit and Engin Erzin for participating in my committee.

I consider myself very lucky to have worked with members of AI Lab. Among them, I would like to especially thank İlker Kesen, Ozan Arkan Can and Omer Kırnap. We learned from each other both academically and socially.

I thank Ekin Akyürek for his valuable helps on this study and Memduh Çağrı Demir for proofreading the thesis.

I thank my family. I always feel their unconditional love with me.

I kept the biggest "thank you" to the end. I would like to thank Seçil million times. Without her support and patience, I doubt I would complete this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

In this thesis, I introduce MorphNet, a sequence-to-sequence recurrent neural network model that takes sentences in plain text as input and attempts to produce the correct morphological analysis of each word as output. Once trained, the model can be used either as a stand-alone application or with an external analyzer to eliminate errors for unambiguous tokens. Traditional methods, e.g. Oflazer and Tür (1997), first produce all possible analyses of a word using finite-state transducers and then perform disambiguation using statistical or rule-based methods. In contrast, MorphNet combines the analysis and disambiguation in a single model and it obtains state-of-the art or comparable results in producing the correct morphological analysis of each word.

Morphological analysis identifies the structure of words and word-parts (morphemes) such as stems, prefixes and suffixes. For example, Table 1.1 shows the three possible analyses for the ambiguous Turkish word "*masalı*": the accusative and possessive forms of the root "*masal*" (tale) and the `+With` form of the root "*masa*" (table) are expressed with the same surface form (Oflazer, 1994). Producing the correct analysis is essential for downstream syntactic and semantic processing.

The importance of morphological analysis varies significantly by language family. Analytic languages like Chinese, and near-analytic languages like English show a low ratio of morphemes to words and express most grammatical relations using function words or word order. According to Baayen et al. (1995), less than 10% of word tokens in English text carry an affix and less than 30 word forms have the type of morphological ambiguity observed in the "*masalı*" example (e.g. *leaves=leaf-f+ves*

Word	**Analysis & Context**
masalı	masal+Noun+A3sg+Pnon+Acc masalı yaz. (write the tale.)
masalı	masal+Noun+A3sg+P3sg+Nom babamın masalı (my father's tale)
masalı	masa+Noun+A3sg+Pnon+Nomˆ DB+Adj+With mavi masalı oda (room with blue table)

Table 1.1: Morphological analysis for Turkish word *masalı*. An example context and its translation is given below each analysis.

vs *leave+s*). Thus, for most purposes, simple table lookup tends to be sufficient for English morphological analysis.

In contrast, agglutinative languages like Turkish and Finnish tend to have a high ratio of morphemes to words and express many grammatical relations using affixes. In the Turkish training data used by Yuret and Türe (2006), 48% of the word tokens carry an affix and 59% of these are morphologically ambiguous. For downstream syntactic analysis, Oflazer et al. (1999) observes that words in Turkish can have dependencies to any one of the inflectional groups of a derived word. For example, in *"mavi masalı oda"* (room with blue table) the adjective *"mavi"* (blue) modifies the noun root *"masa"* (table) even though the final part of speech of "masalı" is an adjective. Accurate morphological analysis and disambiguation are important prerequisites for further syntactic and semantic processing in agglutinative languages.

Morphological analysis is taken to be the task of producing all possible morphological parses of a given word. Morphological analyzers have typically been implemented as Finite State Transducers (FSTs) with language specific, manually generated rules. *Morphological disambiguation* is the task of selecting the correct parse for a given word in a given context among all possible parses. Various rule based, statistical and neural network models have been implemented for morphological disambiguation. These

models are described in Section 2. *Morphological tagging* is the task of assigning a particular part-of-speech and a chain of morphological features to a token in context (Hajič, 2000). In general, lemmatization is not considered as part of morphological tagging (Mueller et al., 2013; Heigold et al., 2017).

In this work, my motivation is to eliminate the need for separate morphological analyzer and disambiguator components and to provide a single, easy-to-use model. This desire can also be interpreted as combining lemmatization and morphological tagging. The model uses three Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) encoders, a character based encoder to obtain word embeddings, a word based bidirectional LSTM encoder to obtain context embeddings and a unidirectional LSTM encoder to obtain output embeddings of preceding word analyses. The decoder consists of a two layer LSTM model. The first layer's hidden state is initialized with the context embedding, and the second layer's hidden state is initialized with the combination of word and output embeddings. The decoder learns to predict the correct morphological analysis including root characters and morphemes.

The structure of this thesis is organized as follows:

Chapter 2 establishes the terminology on morphology and summarizes the related work on Morphological Analysis, Morphological Disambiguation and Morphological Tagging.

Chapter 3 details my model's input output representation and its components.

Chapter 4 describes evaluation datasets in detail and presents results.

Chapter 5 concludes the work in this thesis with a summary of my work.

Appendix contains information about the datasets and model variations used in this thesis.

Chapter 2

RELATED WORK

In this section, I establish terminology on morphology and summarize previous work on Morphological Analysis, Morphological Disambiguation and Morphological Tagging.

2.1 Morphology

Morphology is defined as the study of finding essential linguistic units of language and dividing them into groups based on the behavior and content. Morphology works at the level of word and analyzes the word structure in order to define its syntactic components such as stems, prefixes and suffixes. Besides figuring out structural parts of a word, it also pairs each part with a label from its finite morpheme set. For example, in English the word "cats" consists of two parts "cat" and "-s". The first component is the stem and the second component is an affix indicating plurality. Languages are classified into groups according to the average number of morphemes in a word. Analytic languages are generally considered as low morpheme-per-word ratio languages since they rely on the position of the words in the sentence and auxiliary words such as prepositions rather than the morpheme inflection in order to convey meaning. On the other hand, words in syntactic languages are formed by appending a given number of morphemes to the stem. Syntactic languages are further subgrouped as either agglutinative or fusional. In fusional languages, a morpheme can simultaneously encode several meanings. For example, In Spanish a single morpheme can encode mode, tense, person and aspect feature of the corresponding word. Unlike fusional languages, in agglutinative languages each morpheme represents a single meaning.

2.2 Morphological Analysis

Morphological analysis is generally performed by finite-state transducers (FST) which produce all possible parses for a given word (Koskenniemi, 1983). The analyzers are language dependent rule based systems that typically consist of a lexicon, two-level phonological rules, and finite state transducers that encode morphotactics (Koskenniemi, 1981; Karttunen and Wittenburg, 1983). The first rule based analyzer for Turkish was developed in Oflazer (1994). It was implemented using the Xerox finite state tool (Beesley and Karttunen, 2003) and encoded verbal, nominal and other paradigms as separate finite-state transducers. Figure 2.1 shows the FST diagram that is used by Oflazer (1994) for nominal paradigms. The boxes demonstrate suffixation states and the circles demonstrate final states. The arrows demonstrate the next states which are attainable if a suffix is matched with any of the labels. Other analyzers for Turkish include Eryiğit and Adalı (2004) and Çöltekin (2010). Eryiğit and Adalı (2004) performs morphological analysis by taking affix stripping approach where affixes determined before the root of target word. Çöltekin (2010) is the first freely available two-level morphological analyzer for Turkish. Straka and Straková (2017) proposes a language independent morphological guesser in their pipeline, called UD-Pipe, to perform morphological analysis of CoNLL-U formatted documents. Section 4.1.1 explains CoNLL-U format in detail.

2.3 Morphological Disambiguation

Previous work on morphological disambiguation can be grouped into four broad categories in terms of the techniques they use. These categories are rule based, statistical, hybrid and neural network based approaches.

The rule-based approaches (Karlsson et al., 1995; Oflazer and Kuruöz, 1994; Oflazer and Tür, 1996; Daybelge and Çiçekli, 2007; Daoud, 2009) exploit hand-crafted rules to select the correct parse among the candidates produced by the analyzer. The rules, typically language specific, are designed such that the model can capture the

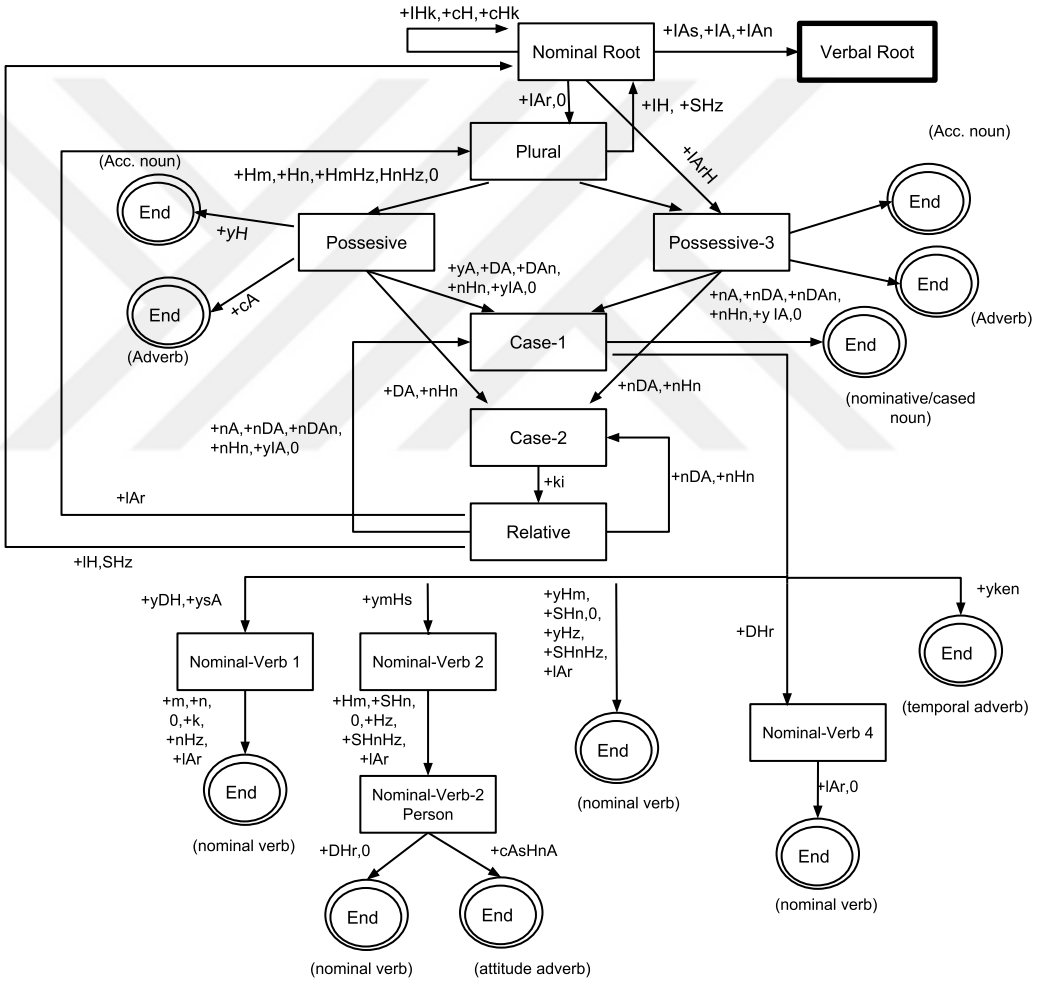

Figure 2.1: Finite State Transducer for nominal paradigm used in Oflazer (1994).

relationship between the context of the word that is subject to disambiguation and all its possible parses.

Statistical approaches try to disambiguate the target word according to the statistics calculated on the corpus they use. For example, Hakkani-Tür et al. (2002) breaks up the morphosyntactic tags into inflectional groups to handle a large set of tags and then the model assigns a probability to each morphosyntactic tag by considering statistics over the individual inflection groups in a trigram model. Yuret and Türe (2006) uses decision lists to vote on each of the potential parses of a word. For each tag in the corpus, a different decision list is learnt using the Greedy Prepend Algorithm (Yuret and de la Maza, 2006).

Hybrid approaches combine hand-crafted rules with statistical methods. For example, Hajič et al. (2007) employs a rule-based component to reduce the number of possible parses and then runs a statistical Part of Speech (POS) tagger to disambiguate words in the Czech language.

Recently, a number of studies (Yıldız et al., 2016; Shen et al., 2016; Toleu et al., 2017) based on neural networks were also tried in morphological disambiguation. Yıldız et al. (2016), proposed an architecture based on a convolutional neural network (CNN). The CNN is used to create a representation of the target word by using the root of the word and its morpheme features. Then by using this representation and ground truth annotations of previous words, the model predicts the correct analysis. In contrast to Yıldız et al. (2016), to capture the context and relationship between target word and its surroundings, Shen et al. (2016) exploits LSTM based neural network architectures. They customize the neural network architecture based on the input language. Toleu et al. (2017) proposes a neural network model for the morphological disambiguation task on Kazakh and Turkish languages. Straka and Straková (2017) employs a supervised, rich feature averaged perceptron to perform disambiguation step in their pipeline. The perceptron disambiguates output of the morphological guesser generating several pairs each of which contains a POS tag and morphological features for each word according to last four characters of the word.

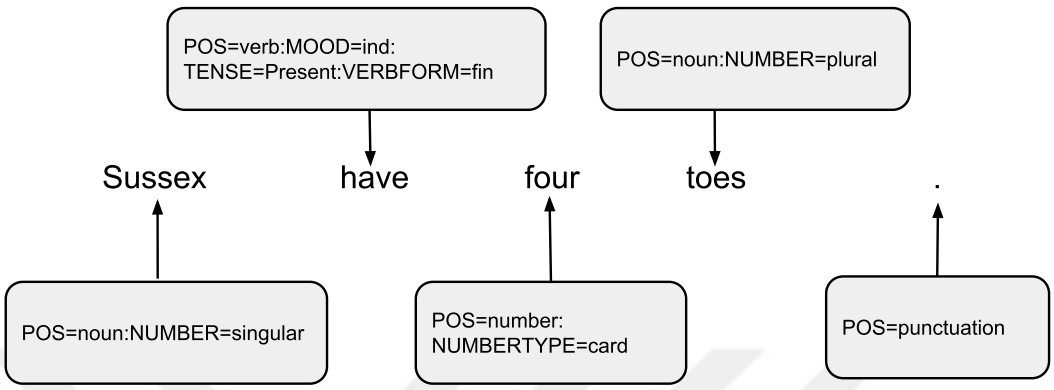

Figure 2.2: Example input output pair for Heigold et al. (2017).

2.4 Morphological Tagging

In contrast to the two stage systems described above which use separate modules for morphological analysis and disambiguation, morphological tagging attempts to solve both problems at once and assign the correct tag to a given word without the aid of an analyzer that produces possible parses. This is the approach I take in my model. Recently, Mueller et al. (2013) attempted to solve the morphological tagging problem by employing a model based on Conditional Random Fields (CRFs) (Lafferty et al., 2001) and Heigold et al. (2017) proposed two morphological tagging architectures based on neural networks. Their best performing architecture, similar to my model, uses unidirectional and bidirectional LSTMs and takes only raw sentences as input. Figure 2.2 shows an example input output pair used in Heigold et al. (2017). My work differs from theirs in a couple of important ways. First, given a word in a context, they predict a morphological tag as a complete unit, rather than as a combination of component features. Consequently, their model can only produce analyses observed in the training data. In contrast, my model produces the morphological tag of the target word a feature at a time. Second, their system ignores the root of the target word and predicts only the morphological features whereas my model generates the stem as well as the morphological tags.

Chapter 3

MODEL

In this chapter, I outline the architecture of my model. First, I define the input/output representation. Next, I explain the neural components of my model. Finally, I outline how I assembled these modules to build my model.

3.1 Input/Output Representation

The input to the model consists of an N word sentence $S = [w_1, \ldots, w_N]$, where w_i is the i'th word in the sentence. Each word is input as a sequence of characters $w_i = [w_{i1}, \ldots, w_{iL_i}], w_{ij} \in \mathcal{A}$ where $\mathcal{A}$ is the set of alphanumeric characters and L_i is the number of characters in word w_i.

The output for each word consists of a stem, a part-of-speech tag and a set of morphological features, e.g. "masal+Noun+A3sg+P3sg+Nom" for "masalı". The stem is produced one character at a time, and the morphological information is produced one feature at a time. A sample output for a word looks like $[s_{i1}, \ldots, s_{iR_i}, f_{i1}, \ldots, f_{iM_i}]$ where $s_{ij} \in \mathcal{A}$ is an alphanumeric character in the stem, R_i is the length of the stem, M_i is the number of features, $f_{ij} \in \mathcal{T}$ is a morphological feature from a feature set such as $\mathcal{T} = \{\text{Noun,Adj,Nom,A3sg}, \ldots\}$.

3.2 Neural Components

In this section, First I describe the individual components of my model and then I present the unified model.

3.2.1 Word Encoder

I map each character w_{ij} to an A dimensional character embedding vector $a_{ij} \in \mathbb{R}^A$.The word encoder takes each word and processes the character embeddings from left to right producing hidden states $[h_{i1}, \ldots, h_{iL_i}]$ where $h_{ij} \in \mathbb{R}^H$. The final hidden state $e_i = h_{iL_i}$ is used as the word embedding for word w_i.

$$h_{ij} = \text{LSTM}(a_{ij}, h_{ij-1}) \tag{3.1}$$

$$h_{i0} = 0 \tag{3.2}$$

$$e_i = h_{iL_i} \tag{3.3}$$

3.2.2 Context Encoder

I use a bidirectional LSTM as the context encoder. The inputs are the word embeddings $e_1, \cdots, e_N$ produced by the word encoder. The context encoder processes them in both directions and constructs a unique context embedding for each target word in the sentence. For a word w_i I define its corresponding context embedding $c_i \in \mathbb{R}^{2H}$ as the concatenation of the forward $\overrightarrow{c}_i \in \mathbb{R}^H$ and the backward $\overleftarrow{c}_i \in \mathbb{R}^H$ hidden states that are produced after the forward and backward LSTMs process the word embedding e_i. In Figure 3.1(c) below, the creation of the context vector for the target word "elini" is illustrated.

$$\overrightarrow{c}_i = \text{LSTM}_f(e_i, \overrightarrow{c}_{i-1}) \tag{3.4}$$

$$\overleftarrow{c}_i = \text{LSTM}_b(e_i, \overleftarrow{c}_{i+1}) \tag{3.5}$$

$$\overrightarrow{c}_0 = \overleftarrow{c}_{N+1} = 0 \tag{3.6}$$

$$c_i = [\overrightarrow{c}_i; \overleftarrow{c}_i] \tag{3.7}$$

3.2.3 Output Encoder

The output encoder captures information about the morphological features of words processed prior to each target word. For example, in order to assign the correct possessive marker to the word "*masalı*" (tale) in "*babamın masalı*" (my father's tale),

it would be useful to know that the previous word "*babamın*" (my father) has a genitive marker. During training I use the gold morphological features for prior words, during testing I use the output of the model. The output encoder only uses the morphological features, not the stem characters, of the previous words as input: $[f_{11}, \ldots, f_{1M_1}, f_{21}, \ldots, f_{i-1,M_{i-1}}]$. I map each morphological feature f_{ij} to a B dimensional feature embedding vector $b_{ij} \in \mathbb{R}^B$. A unidirectional LSTM is run over the whole sentence up to the target word to produce hidden states $[t_{11}, \ldots, t_{i-1,M_{i-1}}]$ where $t_{ij} \in \mathbb{R}^H$. The final hidden state preceding the target word $o_i = t_{i-1,M_{i-1}}$ is used as the output embedding for word w_i.

$$t_{ij} = \text{LSTM}(b_{ij}, t_{ij-1}) \tag{3.8}$$

$$t_{i0} = t_{i-1,M_{i-1}} \tag{3.9}$$

$$o_i = t_{i-1,M_{i-1}} \tag{3.10}$$

3.2.4 Decoder

The decoder is implemented as a 2-Layer LSTM network that outputs the correct tag for a single target word. By conditioning on the input embeddings and its own hidden state, the decoder learns to generate $y_i = [y_{i1}, \ldots, y_{iK_i}]$ where y_i is the correct tag of the target word w_i in sentence S, $y_{ij} \in \mathcal{A} \cup \mathcal{T}$ represents both stem characters and morphological feature tokens, and K_i is the total number of output tokens (stem + features) for word w_i. The first layer of the decoder is initialized with the context embedding c_i.

$$d^1_{i0} = \text{relu}(W_d \times c_i \oplus W_{db}) \tag{3.11}$$

$$d^1_{ij} = \text{LSTM}(y_{ij-1}, d^1_{ij-1}) \tag{3.12}$$

where, $W_d \in \mathbb{R}^{H \times 2H}$, $W_{db} \in \mathbb{R}^H$ and $\oplus$ is element-wise summation. The second layer is initialized with the word and output embeddings after combining them by element-wise summation.

$$d^2_{i0} = e_i + o_i \tag{3.13}$$

$$d^2_{ij} = \text{LSTM}(d^1_{ij}, d^2_{ij-1}) \tag{3.14}$$

I parameterize the distribution over possible morphological features and characters at each time step as

$$p(y_{ij}|d_{ij}^2) = \text{softmax}(W_s \times d_{ij}^2 \oplus W_{sb}) \tag{3.15}$$

where $W_s \in \mathbb{R}^{|\mathcal{Y}| \times H}$ and $W_{sb} \in \mathbb{R}^{|\mathcal{Y}|}$ where $\mathcal{Y} = \mathcal{A} \cup \mathcal{T}$ is the set of characters and morphological features in output vocabulary.

3.2.5 Unified Model

My model is based on the sequence-to-sequence encoder-decoder network approach proposed by Sutskever et al. (2014) for machine translation. I use all of the encoder modules to create embeddings of various input features. First, a word encoder creates an embedding for each word based on its characters. Second, a context encoder creates an embedding for the context of each word based on the word embeddings of its left and right neighbors. Third, an output encoder creates an output embedding using the analyses of previous words. These embeddings are fed to the decoder which produces the stem and the morphological features of a target word. Figure 3.1 gives the model architecture.

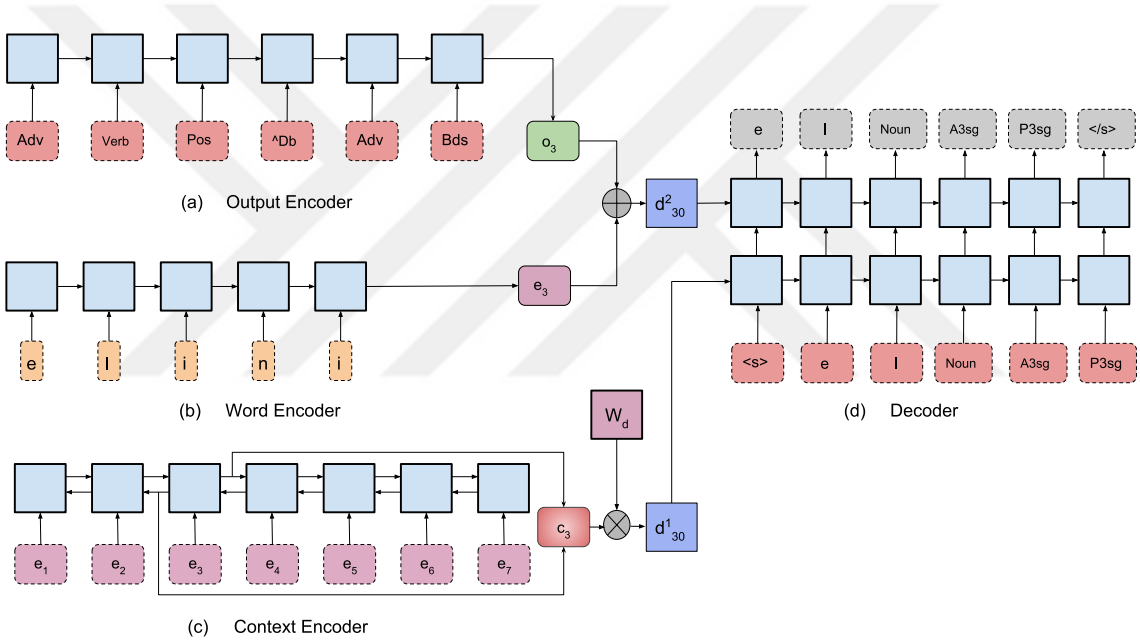

Figure 3.1: Model illustration for the sentence *"Sonra gülerek elini kardeşinin omzuna koydu"* (Then he laughed and put his hand on his brother's shoulder) and target word *"elini"* (his hand). I use the morphological features of the words preceding the target as input to the output encoder: "Sonra+Adv gül+Verb+Pos^DB+Adverb+ByDoingSo".

Chapter 4

EXPERIMENTS

This chapter outlines my experiments. I start by describing the properties of datasets I use. Next, I detail the network training process. Then I describe the transfer learning schema I employ. Finally, I present my model's results.

4.1 Datasets

In this section, I provide information about datasets used for my model evaluation. First I describe the multilingual datasets I used from the Universal Dependencies Treebanks (UD) Nivre et al. (2016). Then, I describe the three morphological disambiguation datasets I used for Turkish.

4.1.1 Universal Dependencies Datasets

I tested my model on a diverse set of languages selected from Universal Dependencies treebanks. UD is a framework for cross-linguistically consistent grammatical annotation and an open community effort with over 200 contributors producing more than 100 tree-banks in over 60 languages. All of the datasets have conventional training, development and test splits. Specifically, I use the CoNLL-U format for the input files. Table 4.1 presents an example CoNLL-U file for English. A CoNLL-U file contains three different line types. *Comment lines* start with # symbol. *Word lines* contain annotations in 10 columns for each token in the file. Lastly, *blank lines* are used to mark the sentence boundaries. I take column 2 (FORM) as input and predict columns 3 (LEMMA), 4 (UPOSTAG) and 6 (FEATS). Additional information about UPOSTAG and FEATS used in each dataset are provided in Appendix B.1 and Appendix B.2. I experiment with several language families. For Indo-European

ID	WORD	LEMMA	UPOSTAG	XPOSTAG	FEATS	HEAD	DEPREL	DEPS	MISC
# text = They buy and sell books									
1	They	they	PRON	PRP	Case=Nom|Number=Plur	2	nsubj	2:nsubj|4:nsubj	_
2	buy	buy	VERB	VBP	Number=Plur|Person=3|Tense=Pres	0	root	0:root	_
3	and	and	CONJ	CC	_	4	cc	4:cc	_
4	sell	sell	VERB	VBP	Number=Plur|Person=3|Tense=Pres	2	conj	0:root|2:conj	_
5	books	book	NOUN	NNS	Number=Plur	2	obj	2:obj|4:obj	SpaceAfter=No
# text = I have no clue									
1	I	I	PRON	PRP	Case=Nom|Number=Sing|Person=1	2	nsubj	_	_
2	have	have	VERB	VBP	Number=Sing|Person=1|Tense=Pres	0	root	_	_
3	no	no	DET	DT	PronType=Neg	4	det	_	_
4	clue	clue	NOUN	NN	Number=Sing	2	obj	_	SpaceAfter=No

Table 4.1: CoNLL-U File format. Word lines containing the annotation of a word/token in 10 fields separated by single tab characters. ID: Word index, integer starting at 1 for each new sentence. FORM: Word form or punctuation symbol. LEMMA: Stem of word form. UPOSTAG: Universal POS tag. XPOSTAG: Language-specific POS tag. FEATS: List of morphological features. HEAD,DEPREL,DEPS: Dependency Parsing related annotations. MISC: Miscellaneous.

(Romance) group, I experiment on Spanish (es), Catalan (ca), Italian (it), French (fr), Greek (el), Portuguese (pt), Romanian (ro) and Hindi (hi). For Germanic family, I experiment on Danish (da), Dutch (nl), Swedish (sv) and English (en). For the Slavic family, I evaluate MorphNet on Bulgarian (bg), Croatian (hr), Czech (cs), Polish (pl), Russian (ru) Slovenian (sl) and Slovak (sk). For Ural-Altaic group, I use Hungarian (hu), Estonian (et), Finnish (fi) and Turkish (tr) languages. For Semitic languages, I experiment on Arabic (ar) and Hebrew (he). Finally, for Sino-Tibetan group, I use Chinese (zh). Table-4.2 summarizes the corpus statistics about each tree-bank.

lang	**train**	**dev**	**test**	**|T|**	**|F|**	**lang**	**train**	**dev**	**test**	**|T|**	**|F|**
ar	230796	31372	29060	363	36	es	384554	37349	12069	404	46
bg	124336	16089	15724	439	45	pt	211820	11158	10468	380	47
cs	1174261	159437	174068	2776	112	sk	80575	12440	13028	1294	61
en	204585	25148	25096	121	35	sl	112530	14063	14077	1140	62
et	22525	12103	12410	599	59	ru	75964	11877	11548	750	39
fr	359477	36101	10090	228	36	hi	281057	35217	35430	985	43
hu	20166	11418	10448	716	73	he	148401	12344	13308	540	48
ro	185113	17074	16324	462	59	el	31212	10139	10422	355	43
tr	38834	10226	10199	1116	67	pl	63013	10332	10897	1047	56
hr	169283	14533	13228	1129	52	sv	66645	9797	20377	211	40
it	277032	12193	10652	278	40	fi	162621	18290	21041	1810	89
da	80378	10332	10023	159	44	nl	186467	11458	10051	335	65
ca	417750	56515	57929	269	55	zh	98608	12663	12012	32	12

Table 4.2: Data statistics of UD Version 2.0 Treebanks. The values in the {train, dev, test} columns are the number of tokens in the splits. $|T|$ gives the number of distinct sets of tags (pos + morphological features) assigned to words, $|F|$ the number of distinct features that exist in the dataset.

4.1.2 Turkish Morphological Analysis/Disambiguation datasets

For Turkish I evaluate my model on three datasets. The first, TrMor2006, was first used in Yuret and Türe (2006). The training set was disambiguated semi-automatically and has limited accuracy. The test set was hand-tagged but is very small (862 tokens) to reliably distinguish between models with similar accuracy. I randomly extracted 100 sentences from training set and used them as the development set while training my model. The first part of Table-4.3 gives number of tokens in each split.

The second dataset I used, TrMor2016, was prepared by Yıldız et al. (2016). The training set is the same with TrMor2006 but they manually re-tagged a subset of the training set containing roughly 20000 tokens to be used as a larger test set. Unfortunately they did not exclude these sentences from the training set in their experiments. Also, they extracted a larger portion of the training set to use as the development set while training. The second part of Table-4.3 gives the statistics for this dataset.

I also evaluate my model on a new dataset, TrMor2018, which I released together with this thesis. TrMor2018 addresses the problems with the accuracy of the training set in TrMor2006 and to provide a larger disjoint test set to more reliably distinguish the accuracy of similar models. The new dataset consists of 34673 sentences and 460663 words in total. Similar to Yuret and Türe (2006), it was annotated semi-automatically in multiple passes. In order to estimate the noise level of the dataset,a randomly selected subset from the dataset was manually disambiguated. The subset contains 2090 sentences and 28909 words. Two annotators annotated each word independently and the final morphological tag of each word was assigned based on the adjudication by a third. Then, I compared the manually disambiguated subset with the semi-automatic results and found the noise level of the dataset as 3% approximately. I splitted TrMor2018 to train, development and test sets by randomly selecting the 80%, 10% and 10% of the sentences respectively. The bottom part of Table-4.3 shows the statistical information about TrMor2018.

	Ambiguous	**Unambiguous**	**Total**
TrMor2006			
Train	397648	438491	836139
Dev	642	743	1385
Test	379	483	862
TrMor2016			
Train	379127	416606	795733
Dev	20096	21695	41791
Test	9460	9802	19262
TrMor2018			
Train	170266	197630	367896
Dev	21479	24645	46124
Test	21477	25166	46643

Table 4.3: Corpus statistic for datasets TrMor2006, TrMor2016 and TrMor2018. Number of tokens for each split in the dataset

4.2 Network Training

All LSTM modules have $H = 512$ hidden units in my model. The size of the character embedding vectors are $A = 64$ in the word encoder. In the decoder part, the size of the output embedding vectors are $B = 256$. The model parameters are initialized with Xavier initialization (Glorot and Bengio, 2010).

The network is trained by using back-propagation through time with stochastic gradient descent. I use learning rate l=1.6 and apply learning rate decay according to development accuracy. I apply learning rate decay by a factor of 0.8 if the development set accuracy is not improved during 3 consecutive epochs and early stopping if the development set accuracy is not improved during 6 consecutive epochs. I apply dropout with the rate of 0.3 before and after each of the LSTM units as well as embedding layers.

4.3 Transfer Learning Formulation

In this section, I introduce what is transfer learning and why it warrants attention in Natural Language Processing. Then I explain the transfer learning formulation I apply and the selection criteria I use to identify the source and target language pairs.

The treebanks in Universal Dependencies differ from each other substantially in terms of the dataset sizes. For example, while there exists more than 1 million morphologically annotated words in Czech treebank, Hungarian and Estonian languages contain 20,166 and 22,525 tokens respectively in their training sets. Table 4.2 shows data statistics for all the languages used in this thesis. Neural methods are data-hungry and struggle in the case of low-resource setting. To mitigate insufficient annotated data problem, I consider transfer learning.

Transfer learning is a technique that gives models opportunity to use knowledge gained from a task on a different but related task. It is an efficient method used in many different Natural Language Processing problems such as Neural Machine Translation (Zoph et al., 2016), Sentiment Analysis (Yoshida et al., 2011) and De-

pendency Parsing (Guo et al., 2016). In order to be successful at transfer learning, the knowledge should be applicable in both source and target sides. In this regard, it is convenient to apply transfer learning on Universal Dependencies treebanks because there is a single, universal annotation set that is shared across all languages although each language may or may not use all of the keys and attributes within the set.

In general, transfer learning can be applied in a couple of different ways. For morphological tagging, one option would be training taggers for both source and target languages jointly by using union of available data for those languages together. This approach requires modifying the input words so that the neural model can distinguish the language to which the words belong. An alternative transfer learning strategy, which I exploit in my thesis, would be training the high-resource language (source) first then use the resulting trained network to initialize and constrain training for low-resource language (target). Unlike the first approach, there is no need to put in extra effort to regulate the input tokens since they are used separately.

Independently of transfer learning strategy being used, it is extremely important to pair languages properly to benefit from transfer learning. In order to identify these pairs, I decide to exploit the treebank statistics for each language. First I count how many times each key in UD annotation set occurs in the treebank and then normalize these values dividing by total number of tokens in the treebank. Next, I create a vector whose length is equal to number of unique keys in UD annotation schema, in order to store these normalized values. I consider each vector as the representative of the corresponding treebank. For each target language, I sort all pairs by measuring the euclidean distance between vectors of the language pair and nominate three closest language as the source language. Finally, I try to transfer knowledge from each source language candidate separately and select the one that improves target language's performance most.

4.4 Results

In this section, I evaluate my model and compare results with related work for Turkish disambiguation datasets and Universal Dependencies datasets.

4.4.1 Turkish Morphological Analysis/Disambiguation Results

Here, I present the performance of my model on three Turkish datasets. Table-4.4 shows the results of several systems for different Turkish datasets. I use per token accuracy as evaluation metric. The *(W/)* column shows the performance when unambiguous tokens are tagged perfectly. The *(W/O)* column shows the accuracies when unambiguous tokens are also tagged by the model. The reason I made this distinction was to get results comparable to older models: When I use my model to tag unambiguous words, it does not achieve 100% accuracy. Unlike my model, all of the models I compare with take the output of morphological analyzer as input and predict the correct analysis among them, thus they always get the right result for unambiguous tokens.

Method	**TrMor2006**			**TrMor2016**			**TrMor2018**		
	A/	**W/**	**W/O**	**A/**	**W/**	**W/O**	**A/**	**W/**	**W/O**
Yuret and Türe (2006)	89.43	95.82	-	-	-	-	-	-	-
Sak et al. (2007)	90.76	96.28	-	-	-	-	-	-	-
Yıldız et al. (2016)	-	-	-	84.12	92.20	-	-	-	-
Shen et al. (2016)	91.03	96.41	-	-	-	-	-	-	-
MorphNet	92.86	96.86	96.63	84.41	92.34	91.93	96.40	98.30	97.69

Table 4.4: Test set performance of several models on Turkish Datasets. A:Ambiguous token accuracy, W:Accuracy on all tokens if unambiguous tokens are tagged perfectly, W/O:Accuracy on all tokens if unambiguous tokens are also tagged with MorphNet.

For the TrMor2006 dataset, I report 96.86% total accuracy which is the best result to date, although the difference is not statistically significant. For ambiguous words, my model performs with 92.86% accuracy showing its ability to simulate both analyzer and disambiguator internally. Maybe more importantly, the standalone version of my model (W/O) performs better than the disambiguators even though it does not rely on an external analyzer. When I test my model on TrMor2016 dataset, it achieved a slightly better performance than Yıldız et al. (2016) on ambiguous words. I hope that the new TrMor2018 dataset will allow for better system comparison due to its high accuracy and large test set.

4.4.2 Universal Dependencies Morphological Tagging Results

Here I present the performance of my model on Universal Dependencies datasets on UPOS tagging, FEAT tagging and lemmatization. I used a single model to produce the lemma, UPOS tag and FEAT tags of each input word. After training, I split my model's output in order to evaluate performance of it on these three tasks separately. To prove MorphNet is a language agnostic model and it can easily be used with languages other than Turkish I evaluated my model on 26 different languages and compared results with previous work which report the top scores on UD tree-banks.

Straka and Straková (2017) use a morphological guesser and average structured perceptron with 38 different hand-crafted features to build a tagger. Their guesser generates several options for each word according to its suffixes with different lengths, and then the averaged perceptron tagger disambiguates the generated tags. Their scores are taken from Straka and Straková (2017). Heigold et al. (2017) employs neural networks to build their morphological tagger. They produce a composite tag for each word by utilizing a bidirectional LSTM over a character based word vectors. Heigold et al. (2017) provide scores on Universal Dependencies(UD) v1.3 datasets.

To compare my model with all previous work simultaneously, I re-implemented the model with parameters in Heigold et al. (2017) and trained on UD V2.0 datasets. For Hungarian (hu), Estonian (et), Chinese (zh), Greek (el), English (en),French (fr)

and Italian (It) I use my model, described in Chapter 3, with minor variations. See Appendix A.1 for details. Table-4.5, Table-4.6 and Table-4.7 present the results for UPOS tagging, FEAT tagging and lemmatization. Moreover, Appendix C.2 shows my model's development set results on several CoNLL 2018 datasets.

4.4.2.1 Universal Part-Of-Speech (UPOS) Tagging Results

Here I focus on the UPOS tagging performance of my model and previous work. I compared my model to Heigold et al. (2017) and Straka and Straková (2017). In this section, I ignored the correctness of my model's lemma and FEATs predictions and consider its UPOS prediction only. The results are presented on Table-4.5. The scores on the table proves that my model performs robustly across wide range of languages and it achieves similar or better scores than the previous work in UPOS tagging on Universal Dependencies treebanks. My model performs best for most of the agglutinative languages such as Turkish (tr) and Finnish (fi). For Turkish MorphNet achieves 95.11% accuracy which is considerable better than the previous best. The Hungarian (hu) and Estonian (et) are the exceptions where Straka and Straková (2017) performs significantly better than both my model and Heigold et al. (2017). Straka and Straková (2017) moves sentences from the beginning of development set to the end of training set, until the training set is at least nine times the development set. This strategy effects the datasets mostly on these languages because both Hungarian and Estonian datasets have the lowest train set/ development set ratio and applying this method increases the sizes of Hungarian and Estonian training sets up to 50%. For some Slavic languages such as Bulgarian (bg) and Slovenian (sl), my model performs slightly better than Straka and Straková (2017) while Heigold et al. (2017) achieves best scores by additional minor gains. My model performs equally with Heigold et al. (2017) on Czech (cs) by 98.58% accuracy and Straka and Straková (2017) on Russian (ru) by 94.80% accuracy. For Dutch (nl), my model performs significantly worse than previous work. The performance of my model on languages such as Arabic (ar), Chinese (zh), Hebrew (he) and Hindi (hi), where it performs best with 94.89%,

lang	MorphNet	Heigold et al	UD-Pipe	lang	MorphNet	Heigold et al	UD-Pipe
ar	94.89	94.45	94.40	es	95.85	96.40	95.92
bg	97.93	98.01	97.70	pt	95.07	97.04	96.80
cs	98.58	98.58	98.40	sk	91.82	92.30	93.30
en	94.01	94.12	94.50	sl	96.24	96.74	96.20
et	86.73	83.71	91.30	ru	94.80	94.59	94.80
fr	95.60	95.50	96.50	hi	96.21	96.07	95.99
hu	87.36	78.47	91.80	he	95.51	94.78	95.10
ro	96.74	96.84	96.80	el	95.10	93.68	96.00
tr	95.11	93.35	94.00	pl	95.48	94.26	95.70
hr	96.18	96.27	96.00	sv	95.01	95.52	95.80
it	97.85	97.33	97.40	nl	85.77	91.98	91.80
da	95.45	95.54	95.50	fi	95.48	95.33	94.90
ca	96.42	98.25	98.00	zh	92.90	91.12	92.20

Table 4.5: Test set UPOS performances. Values are per-token accuracies. The MorphNet column presents my model's results on UPOS tagging.

92.90%, 95.51% and 96.21% accuracies respectively, proves that MorphNet is not affected by the character set of the input language. Finally, For English (en) and French (fr) Straka and Straková (2017) performs best as a result of employing a morphological guesser whose feature-set is fine tuned on English and French datasets.

4.4.2.2 Morphological Feature (FEAT) Tagging Results

In this section, I present my model's performance on FEAT Tagging and compare it with Heigold et al. (2017) and Straka and Straková (2017). The results are presented in Table-4.6. While calculating the accuracy, I consider only the correctness of FEATS part of my model's output and ignore the lemma and UPOS parts. There are 48 unique feature keys and 177 unique key-value pairs in the Universal Dependencies annotation. Each tag contains a chain of keys for FEAT tagging. It makes FEAT tagging a more difficult task than UPOS tagging and thus the performance difference between models become more apparent in general. Heigold et al. (2017) predicts the tag as a complete unit rather than as a combination of key features. It performs slightly better than my system for languages with relatively less unique tags such as Catalan (ca) and English (en). On the contrary, my model obtains best results on languages having a large number of unique tags in their corpus. For Finnish (fi) whose corpus contains 1810 unique tags, my model achieves 93.08 and for Czech (cs), where 2776 unique tags exist, it performs with 94.65% FEAT tagging accuracy. Similarly, it improves previous best score for Turkish and Hungarian which are the languages with more than 700 unique tags, by 3.02% and 4.62% additional gains respectively. For Iberian-Romance languages such as Spanish (es) and Portuguese(pt), Heigold et al. (2017) obtains best results. For Russian and Polish, which are the members of Slavic language family with highly fusional morphology, my model achieves 88.46% and 86.30% accuracies respectively and outperforms previous work by large margins. For another Slavic language, Slovenian (sl), my model performs better than Straka and Straková (2017) by 1.96% while Heigold et al. (2017) achieves the best score by additional 1.67%. While my model outperforms previous work on Hebrew (he) with 92.32% accuracy, Straka and Straková (2017) performs best with 85.00% accuracy on Estonian (et). Similarly, it achieves best results for Swedish (sv) with 94.60% accuracy and for French (fr) with 96.50% accuracy. Finally, for Chinese (zh) and Danish (da) all of the models perform almost equally.

lang	MorphNet	Heigold et al	UD-Pipe	lang	MorphNet	Heigold et al	UD-Pipe
ar	91.48	91.96	89.60	es	95.48	96.40	96.30
bg	96.65	96.53	95.60	pt	94.63	95.41	93.70
cs	94.65	94.32	92.20	sk	79.00	81.30	79.90
en	96.16	96.17	95.40	sl	90.56	92.23	88.60
et	81.75	79.45	85.00	ru	88.46	85.95	84.50
fr	96.33	96.33	96.50	hi	92.01	92.14	90.30
hu	75.22	63.29	70.60	he	92.32	91.77	91.30
ro	96.20	96.34	96.30	el	90.42	89.20	90.50
tr	91.92	88.57	88.90	pl	86.30	85.76	84.20
hr	87.17	87.53	84.40	sv	93.36	94.41	94.60
it	97.72	97.43	97.20	nl	85.21	90.24	89.90
da	94.53	94.55	94.50	fi	93.08	92.81	91.80
ca	95.70	97.60	97.20	zh	98.83	98.73	98.79

Table 4.6: Test set FEAT performances. Values are per-token accuracies. The MorphNet column presents my model's results on FEAT tagging.

4.4.2.3 Lemmatization Results

In this section, I focus on my model's lemmatization performance. I consider only the lemma part of the prediction generated by my model and ignore the UPOS and FEATs sections of it. I compare my model only by Straka and Straková (2017) because, the model presented in Heigold et al. (2017) produces only UPOS and FEAT tags and does not generate a lemma for input tokens. I use exact matching accuracy as the evaluation metric. The results are presented in Table -4.7. Unlike my model, Straka and Straková (2017) use separate models for UPOS+FEAT tagging and lemmatization. Similar to their morphological tagger, Straka and Straková (2017) use a morphological guesser and averaged perceptron tagger for lemmatizer. The guesser is basically a heuristic. It produces several lemma rules based on the last four characters of a word. Next, each lemma rule generates a lemma from a word by stripping some prefix and suffix and prepending and appending new prefix and suffix. Then, an averaged perceptron performs disambiguation to decide the correct lemma form. My model outperforms Straka and Straková (2017) by a considerable margin for 13 languages and performs slightly better for 8 languages. Among the remaining 5 languages, Chinese (zh) and Hungarian (hu) are the only two languages for which my model performs significantly worse than Straka and Straková (2017). Chinese is a non-inflectional language and thus words always appear in the lemma form. The reason why my model does not perform with perfect accuracy is the errors occurred while copying characters of the words. Similar to UPOS and FEAT tagging tasks, %3.05 accuracy difference on Hungarian dataset shows that Straka and Straková (2017)'s strategy of transferring data from the development set to the training shows its effect best on Hungarian language.

lang	**MorphNet**	**UD-Pipe**	**lang**	**MorphNet**	**UD-Pipe**
ar	94.48	92.60	es	98.45	96.10
bg	97.53	94.70	pt	97.78	97.20
cs	98.95	97.90	sk	91.76	86.00
en	98.20	96.90	sl	96.52	95.40
et	84.20	84.50	ru	95.96	75.10
fr	98.16	97.60	hi	98.27	98.00
hu	86.45	89.50	he	93.89	93.20
ro	97.72	96.80	el	94.23	94.60
tr	96.16	91.70	pl	95.74	93.60
hr	96.12	94.40	sv	94.92	95.70
it	98.02	97.50	nl	90.50	90.10
da	96.08	95.00	fi	93.80	86.80
ca	98.15	97.90	zh	95.81	100.00

Table 4.7: Test set lemmatization performances.Values are per-token accuracies. The MorphNet column presents my model's results on lemmatization.

4.4.3 Transfer Learning on Universal Dependencies Datasets

In this section, I present the scores I obtain after applying transfer learning. The results are presented in Table-4.8. Similar to Section 4.4.2, I report results for UPOS Tagging, FEAT tagging and Lemmatization separately. For each target language in the Table-4.8, I tried to transfer knowledge from its three closest pairs. I reported the best score I obtained for each target language. For complete list of (target,source) pairs, see Appendix C.1. The *Source Lang* column shows the language from whose model I copy the learned weights to initialize the *Target Lang*'s model parameters. The values within parenthesis show the increase (or decrease) I obtain by transfer learning compared to from scratch training. In general, transfer learning shows its effect best when the available corpus for target language is limited. For Estonian (et), which has the smallest training data, I observed at least 2.5% increase in each category when it is initialized with the weights of Finnish model. Similarly, for Hungarian (hu), the lemmatization performance increases by 3.15% and UPOS tagging performance increases by 1.40% but transfer learning hurts FEAT tagging performance approximately 1.00%. When I initialize the weights of Slovak (sk) model with the Czech (cs) model, I obtain 5.20% increase in FEAT tagging accuracy and achieve the best results by outperforming previous work in all of the categories. Initializing Swedish (sv) model with Danish (da), an another Continental Scandinavian language, increases my model's accuracy by at 1.58% in UPOS tagging and 4.34% in FEAT tagging and enables my model to perform slightly better than all previous work. When I swap the source and target for (sv,da) pair, I observe the decrease in the performance increase. I believe it is because Danish (da) have larger training set and therefore it is more applicable to be used as *Source Language*. When I try transfer learning on languages who have large training set such as Portuguese (pt), Hindi (hi) and Romanian (ro), I observed minor performance increases in general. Finally, Spanish model improves the Dutch model's UPOS and FEAT tagging performance more than 1.0 % and Croatian (hr) model helps Slovenian model's FEAT tagging by providing 2.5% gain.

Target Lang	**MorphNet (UPOS)**	**MorphNet (FEAT)**	**MorphNet (Lemma)**	**Source Lang**
et	89.23 (+ 2.50)	85.25 (+3.50)	87.62 (+3.42)	fi
sk	93.74 (+ 1.92)	85.20 (+ 5.20)	94.42 (+ 2.65)	cs
hu	88.75 (+ 1.40)	74.20 (- 1.02)	89.60 (+ 3.15)	fi
da	95.75 (+ 0.60)	94.94 (+ 0.41)	96.46 (+ 0.38)	sv
pt	95.40 (+ 0.33)	95.01 (+ 0.38)	98.03 (+ 0.25)	es
hi	96.44 (+ 0.23)	92.16 (+ 0.15)	98.47 (+ 0.20)	ur
sl	96.91 (+ 0.67)	93.06 (+ 2.50)	97.46 (+ 0.94)	hr
ro	96.70 (- 0.04)	96.53 (+ 0.33)	97.91 (+ 0.19)	bg
nl	86.77 (+ 1.00)	86.53 (+ 1.32)	91.11 (+ 0.61)	es
sv	95.82 (+ 1.58)	94.90 (+ 4.34)	96.72 (+ 0.20)	da

Table 4.8: Test set performances after Transfer Learning. The values within parentheses indicates the improvements in accuracy achieved by transfer learning.

Chapter 5

CONCLUSION

In this thesis, I described a language independent neural sequence-to-sequence model for morphological Disambiguation and Tagging problems. In its most generic form, my model employs three LSTM encoders, a character based encoder to obtain word embeddings, a word based bidirectional LSTM encoder to obtain context embeddings and a unidirectional LSTM encoder to obtain output embeddings of preceding word analyses. It uses a two layer LSTM decoder to predict the correct morphological analysis. I showed that such a model can disambiguate input tokens without relying on an external analyzer. Next, I evaluated my model's lemmatization, Part-of-Speech Tagging and Morphological Feature Tagging performance on Universal Dependencies datasets. Furthermore, I showed that, by transfer learning, I can improve my model's performance even further especially for the low resource languages. By obtaining State-of-the-Art or comparative results on 26 different languages, I demonstrate that my model can achieve good performance on multiple languages with different characteristics.

Appendices

Appendix 1

MODEL

A.1 Model Variations

For a subset of tree-banks in Table-4.2, I use two different variations of my model. For Hungarian, Estonian, Greek and Chinese I used the model without the output encoder. Figure A.1 represents the corresponding variation. The aim was to decrease to model complexity (at the risk of losing its representation capacity) in order to ease the over-fitting. For English, French and Italian I use only word encoder and context encoder by discarding output encoder and decoder. The model predicts the tag of each word compositely and it is trained by using CRF loss. This variation increases my UPOS tagging performance for aforementioned languages even-though it did not effect FEAT tagging significantly. Figure A.2 represents the corresponding variation.

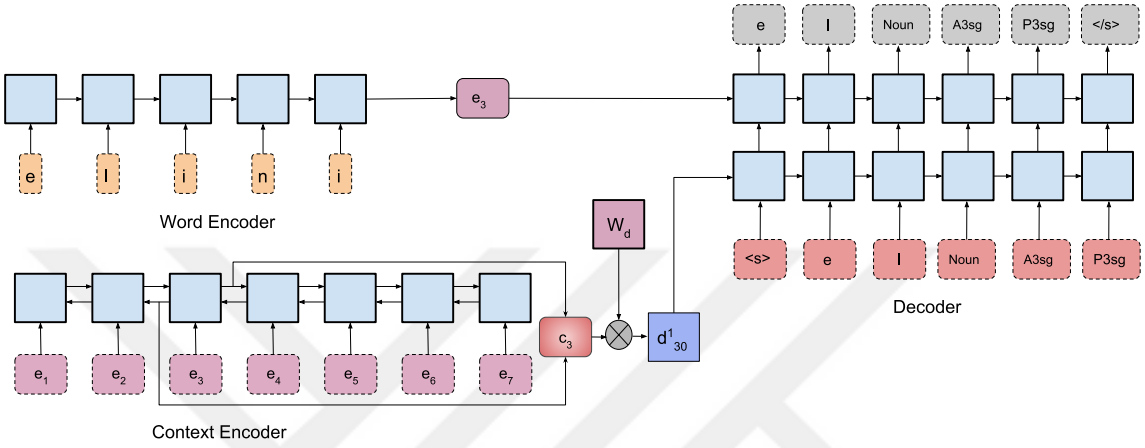

Figure A.1: The first variation of MorphNet.

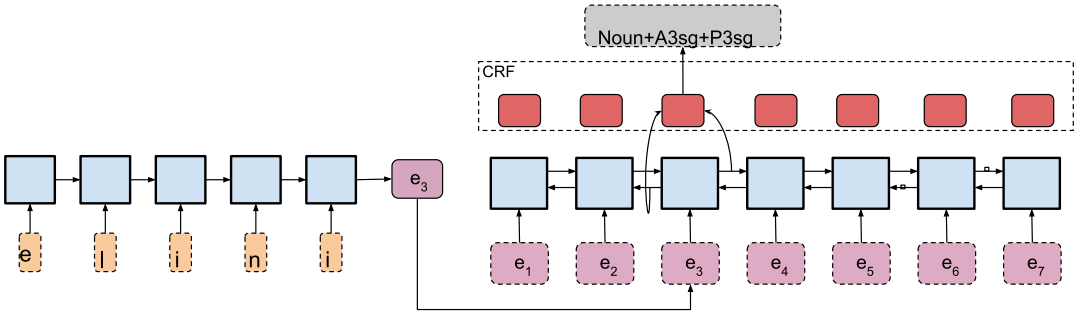

Figure A.2: The second variation of MorphNet.

Appendix 2

DATASETS

B.1 Universal Dependencies POS tags

In this section, I provide the Universal Part-of-Speech (UPOS) tags used in Universal Dependencies (UD) datasets. Table B.1 demonstrates the set of all UPOS tags used in UD V2.0 datasets.

ADJ	Adjective
ADP	Adposition
ADV	Adverb
AUX	Auxiliary
CCONJ	Coordinating Conjunction
DET	Determiner
INTJ	Interjection
NOUN	Noun
NUM	Numeral
PART	Particle
PRON	Pronoun
PROPN	Proper Noun
PUNCT	Punctuation
SCONJ	Subordinating Conjunction
SYM	Symbol
VERB	Verb
X	Other

Table B.1: Universal Dependencies v2.0 Part-of-Speech Tags

B.2 Universal Dependencies Morphological Feature tags

In this section, Morphological Feature (FEAT) tags used in Universal Dependencies datasets. Table B.2 demonstrates the set of all FEAT tags used in UD V2.0 datasets.

Abbr	Abbreviation
AbsErgDatNumber	number agreement with absolutive/ergative/dative argument
AbsErgDatPerson	person agreement with absolutive/ergative/dative argument
AbsErgDatPolit	politeness agreement with absolutive/ergative/dative argument
AdpType	Adposition type
AdvType	Adverb type
Animacy	Animacy
Aspect	Aspect
Case	Case
Clusivity	Clusivity
ConjType	Conjunction type
Definite	Definitiness or state
Degree	Degree of comparison
Echo	Is this an echo word or a reduplicative?
ErgDatGender	Gender aggrement with ergative/dative argument
Evident	Evidentiality
Foreign	Is this a freign word?
Gender	Gender
Hyph	Hyphenated compound or part of it
Mood	Mood
NameType	Type of named entity
NounType	Noun type
NumForm	Numeral form
NumType	Numeral type
NumValue	Numeric value

Number	Number
PartType	Particle type
Person	Person
Polarity	Polarity
Polite	Politeness
Poss	Possessive
PossGender	Possessor's gender
PossNumber	Possessor's number
PossPerson	Possessor's person
PossedNumber	Possessed object's number
Prefix	Word functions as a prefix in a compound construction
PrepCase	Case form sensitive to prepositions
PronType	Pronominal type
PunctSide	Which side of paired punctuation is this?
PunctType	Punctuation type
Reflex	Reflexive
Style	Style or sublanguage to which this word form belongs
Subcat	Subcategorization
Tense	Tense
Typo	Is this a misspelled word?
VerbForm	Form of verb or deverbative
VerbType	Verb type
Voice	Voice

Table B.2: Universal Dependencies v2.0 Morphological Features

Appendix 3

RESULTS

C.1 Target-Source Language Pairs

Here, I list three closest languages for each language that I use as *Target Language* in Table 4.8. I define the distance between languages as a root mean square distance of the corresponding vectors which are the FEAT distributions in the corpus.

Target Lang	**Source Language Candidates**
Estonian	Finnish, Turkish, Hungarian
Slovak	Polish, Czech, Croatian
Hungarian	Latvian, Croatian, Finnish
Danish	Swedish,English,Norwegian
Portuguese	French, Spanish, Galician
Hindi	Urdu, Portuguese, Estonian
Slovenian	Croatian, Czech, Russian
Romanian	Galician, Bulgarian, Slovenian
Dutch	Spanish, Catalan, Romanian
Swedish	Norwegian, Danish, Romanian

Table C.1: Transfer Learning (Target-Source) Language Candidates

C.2 CoNLL 2018 Shared Task Results

Here, I present my model's results obtained on several CoNLL 2018 Shared Task treebanks. Since the test sets have not been released yet here I report development-set results. These results are obtained by starting training from scratch.

lang	**UPOS**	**FEATS**	**LEMMA**	**lang**	**UPOS**	**FEATS**	**LEMMA**
ar	96.14	91.44	94.27	tr	94.46	91.25	95.91
da	95.40	93.70	95.80	el	95.21	90.11	94.27
fi	94.53	95.18	95.27	hr	96.58	89.61	96.51
cs	97.69	92.10	98.15	hu	89.79	70.23	86.52
gl	96.97	99.81	97.70	sl	96.85	90.96	97.08
de	93.09	88.30	96.83	pl	95.88	87.33	95.97
ja	97.43	99.95	97.80	it	94.90	95.34	96.35
sv	94.65	93.80	95.89	hi	96.20	91.71	98.36
bg	97.77	96.65	97.63	ro	97.08	96.52	97.66
zn	91.70	98.82	96.40	sr	96.40	92.77	95.65
grc	91.05	88.68	82.92	af	95.39	93.44	95.41
ko	93.21	98.59	89.91	ur	90.07	86.97	88.76
eu	94.76	91.50	96.48	fr	96.73	96.38	97.80
fro	94.17	96.21	100.00	he	96.35	93.57	96.54
es	97.10	96.66	98.73	ru	97.70	95.97	98.16

Table C.2: MorphNet's development set results on several UD v2.2 datasets that are used in CoNLL 2018 Shared Task.

BIBLIOGRAPHY

Baayen, R. H., Piepenbrock, R., and Gulikers, L. (1995). The celex lexical database (release 2). *Distributed by the Linguistic Data Consortium, University of Pennsylvania*.

Beesley, K. and Karttunen, L. (2003). Finite-state morphology.

Çöltekin, c. (2010). A freely available morphological analyzer for turkish. In *LREC*, volume 2, pages 19–28.

Daoud, D. (2009). Synchronized morphological and syntactic disambiguation for arabic. *Advances in Computational Linguistics*, 41:73–86.

Daybelge, T. and Çiçekli, I. (2007). A rule-based morphological disambiguator for turkishâĂİ, in. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2007), Borovets*, pages 145–149.

Eryiğit, G. and Adalı, E. (2004). An affix stripping morphological analyzer for turkish. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, Innsbruck, Austria*, pages 299–304.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Guo, J., Che, W., Yarowsky, D., Wang, H., and Liu, T. (2016). A representation learning framework for multi-source transfer parsing.

Hajič, J., Votrubec, J., Krbec, P., Květoň, P., et al. (2007). The best of two worlds: Cooperation of statistical and rule-based taggers for czech. In *Proceedings of the*

Workshop on Balto-Slavonic Natural Language Processing: Information Extraction and Enabling Technologies, pages 67–74. Association for Computational Linguistics.

Hajič, J. (2000). Morphological tagging: Data vs. dictionaries. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 94–101, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hakkani-Tür, D. Z., Oflazer, K., and Tür, G. (2002). Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, 36(4):381–410.

Heigold, G., ünter Neumann, G., and van Genabith, J. (2017). An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 1, pages 505–513.

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

Karlsson, F., Voutilainen, A., Heikkila, J., and Anttila, A., editors (1995). *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Walter de Gruyter & Co., Hawthorne, NJ, USA.

Karttunen, L. and Wittenburg, K. (1983). A two-level morphological analysis of english. In *Texas Linguistic Forum Austin, Tex.*, number 22, pages 217–228.

Koskenniemi, K. (1981). An application of the two-level model to Finnish. *Computational morphosyntax: Report on research*, 1984:19–41.

Koskenniemi, K. (1983). Two-level model for morphological analysis. In *IJCAI*, volume 83, pages 683–685.

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

Mueller, T., Schmid, H., and Schütze, H. (2013). Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA. Association for Computational Linguistics.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R. T., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D. (2016). Universal dependencies v1: A multilingual treebank collection. In *LREC*.

Oflazer, K. (1994). Two-level description of turkish morphology. *Literary and Linguistic Computing*, 9(2):137–148.

Oflazer, K., Hakkani-Tür, D. Z., and Tür, G. (1999). Design for a turkish treebank. In *Proceedings of the Workshop on Linguistically Interpreted Corpora, EACL 99*, Bergen, Norway.

Oflazer, K. and Kuruöz, İ. (1994). Tagging and morphological disambiguation of turkish text. In *Proceedings of the fourth conference on Applied natural language processing*, pages 144–149. Association for Computational Linguistics.

Oflazer, K. and Tür, G. (1996). Combining hand-crafted rules and unsupervised learning in constraint-based morphological disambiguation. *CoRR*, cmp-lg/9604001.

Oflazer, K. and Tür, G. (1997). Morphological disambiguation by voting constraints. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL97, EACL97)*, Madrid, Spain.

Sak, H., Güngör, T., and Saraçlar, M. (2007). *Morphological Disambiguation of Turkish Text with Perceptron Algorithm*, pages 107–118. Springer Berlin Heidelberg, Berlin, Heidelberg.

Shen, Q., Clothiaux, D., Tagtow, E., Littell, P., and Dyer, C. (2016). The role of context in neural morphological disambiguation. In *COLING*, pages 181–191. ACL.

Straka, M. and Straková, J. (2017). Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Toleu, A., Tolegen, G., and Makazhanov, A. (2017). Character-aware neural morphological disambiguation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 666–671.

Yıldız, E., Tirkaz, c., Şahin, H. B., Eren, M. T., and Sönmez, O. (2016). A morphology-aware network for morphological disambiguation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2863–2869. AAAI Press.

Yoshida, Y., Hirao, T., Iwata, T., Nagata, M., and Matsumoto, Y. (2011). Transfer learning for multiple-domain sentiment analysis — identifying domain dependent/independent word polarity. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, pages 1286–1291. AAAI Press.

Yuret, D. and de la Maza, M. (2006). The greedy prepend algorithm for decision list induction. In *International Symposium on Computer and Information Sciences*, pages 37–46. Springer.

Yuret, D. and Türe, F. (2006). Learning morphological disambiguation rules for turkish. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 328–334. Association for Computational Linguistics.

Zoph, B., Yuret, D., May, J., and Knight, K. (2016). Transfer learning for low-resource neural machine translation. *CoRR*, abs/1604.02201.